TelidControl

(TELID® library)

Product:	microsensys iID®3000 PRO RFID interfaces TELID®300 sensor data loggers
Product Code:	**-**
Revision:	**0.9.3**
Date:	2021-11-02

API - Definition for Windows® PC, macOS and Linux

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

This documentation describes the software API valid for many microsensys RFID interfaces supporting TELID® sensor logger functionalities.

Class name:	**TELIDLibrary.TelidControl**
Tested devices:	MICROSOFT® Windows 7/64bit, 8.1/32bit/64bit, 10/64bit
Supported host interface:	RS232 serial, USB, Bluetooth SPP
Version:	0.9.3
Release date:	2021-11-02
Dependencies:	.NETStandard v2.0, iIDReaderLibrary v1.0.2
NuGet Package-ID:	Microsensys.TELIDLibrary
NuGet-Link:	https://www.nuget.org/packages/Microsensys.TELIDLibrary/

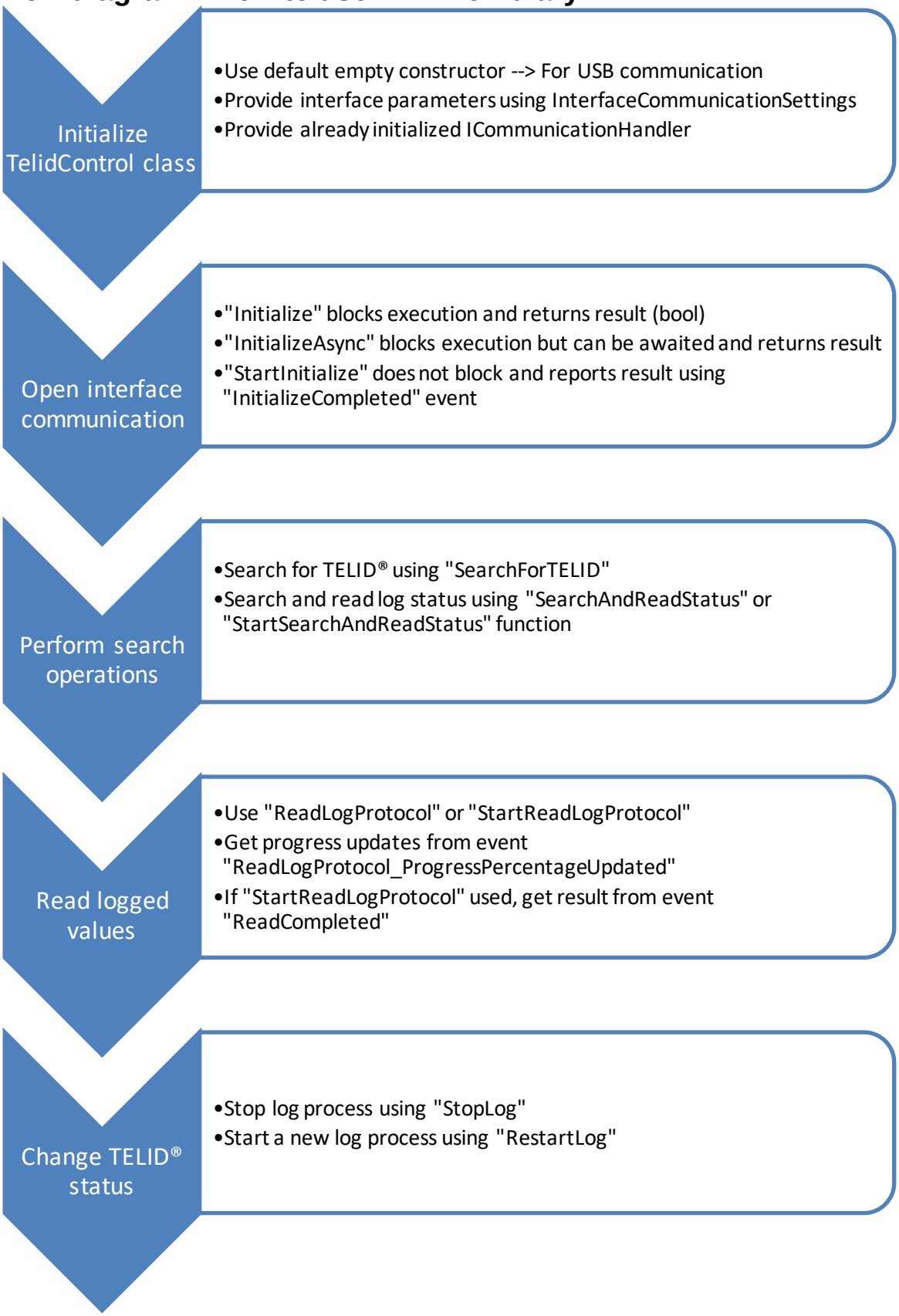
Content

Short history

This chapter includes a short history of modifications of TelidControl.

Date	Reason	Modification	Release Date / Version	FileName
2021-03	- first release		0.9	
2021-11	- implementation of password protected TELID300.nfc loggers	Added “configuration” section in API documentation	0.9.3	

Flow diagram – How to use TELID® 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 TelidControl ()

Parameter :
-
Result :
success – instance of *TelidControl*
error - Exception
Description :
Initializes *TelidControl* class using USB as peripheral interface and automatically checking cyclically if the communication with reader is still possible in the background when no read process is running.

public TelidControl (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 *TelidControl*
error - Exception
Description :
Initializes *TelidControl* class using peripheral interface parameters and automatically checking cyclically if the communication with reader is still possible in the background when no read process is running.

public TelidControl (ICommunicationHandler _commHandler)

Parameter :
_commHandler – instance of *ICommunicationHandler* interface representing the host-side peripheral interface to communicate with (already initialized)
Result :
success – instance of *TelidControl*
error - Exception
Description :
Initializes *TelidControl* class using an already initialized instance of *ICommunicationHandler* and automatically checking cyclically if the communication with reader is still possible in the background when no read process is running.

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

Class 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 *DocInterfaceControl* that communicates using the same host-side peripheral interface.

public int ReaderID

Accessors :
get / -
Description :
Returns the Reader information read using the configured interface.

public int AutoOffValue

Accessors :
get / set
Description :
Returns the last set AutoOff value or sets a new value to the reader. This value configures the time the RFID field stays on after a successful communication process.

Common functions

Search functions

This chapter describes functions available for searching and reading the status of TELID® data logger.

public TelidInformation SearchForTELID ()

Parameter :
-
Result :
success – returns instance of *TelidInformation* representing information of found TELID® logger.
error – returns *null*.
Description :
Searches for a TELID® logger near the antenna. This function blocks the execution until the answer is received or for a maximum of 1000ms.

public TelidInformation SearchForTELID (int _maxScanTimeMs)

Parameter :
_maxScanTimeMs – maximum time in milliseconds that the function will try to find a TELID®.
Result :
success – returns instance of *TelidInformation* representing information of found TELID® logger.
error – returns *null*.
Description :
Searches for a TELID® logger near the antenna. This function blocks the execution until the answer is received or for a maximum of time provided by parameter.

public async Task<TelidInformation> SearchForTELIDAsync ()

Parameter :
-
Result :
success – returns instance of *TelidInformation* representing information of found TELID® logger.
error – returns *null*.
Description :
Searches asynchronously for a TELID® logger near the antenna. This function blocks the execution until the answer is received or for a maximum of 1000ms.

public async Task<TelidInformation> SearchForTELIDAsync (int _maxScanTimeMs)

Parameter :
_maxScanTimeMs – maximum time in milliseconds that the function will try to find a TELID®.
Result :
success – returns instance of *TelidInformation* representing information of found TELID® logger.
error – returns *null*.
Description :
Searches asynchronously for a TELID® logger near the antenna. This function blocks the execution until the answer is received or for a maximum of time provided by parameter.

Search and read status functions

This chapter describes functions available for searching and reading the status of TELID® data logger.

public TelidStateInformation SearchAndReadStatus ()

Parameter :
-
Result :
success – returns instance of *TelidStateInformation* representing information and logging status of found TELID® logger.
error – returns *null*.
Description :
Searches for a TELID® logger near the antenna and reads its current logging status. This function blocks the execution until the answer is received or for a maximum of 1000ms.

public TelidStateInformation SearchAndReadStatus (int _maxScanTimeMs)

Parameter :
_maxScanTimeMs – maximum time in milliseconds that the function will try to find a TELID®.
Result :
success – returns instance of *TelidStateInformation* representing information and logging status of found TELID® logger.
error – returns *null*.
Description :
Searches for a TELID® logger near the antenna and reads its current logging status. This function blocks the execution until the answer is received or for a maximum of time provided by parameter.

public async Task<TelidStateInformation> SearchAndReadStatusAsync ()

Parameter :
-
Result :
success – returns instance of *TelidStateInformation* representing information and logging status of found TELID® logger.
error – returns *null*.
Description :
Searches asynchronously for a TELID® logger near the antenna and reads its current logging status. This function blocks the execution until the answer is received or for a maximum of 1000ms.

public async Task<TelidStateInformation> SearchAndReadStatusAsync (int _maxScanTimeMs)

Parameter :
_maxScanTimeMs – maximum time in milliseconds that the function will try to find a TELID®.
Result :
success – returns instance of *TelidStateInformation* representing information and logging status of found TELID® logger.
error – returns *null*.
Description :
Searches asynchronously for a TELID® logger near the antenna and reads its current logging status. This function blocks the execution until the answer is received or for a maximum of time provided by parameter.

public void StartSearchAndReadStatus ()

Parameter :

-

Result :

-

Description :

Starts a *SearchAndReadStatus* process in a background thread. It searches for a TELID® logger near the antenna and reads its current logging status. This function completes immediately after being call, and results are provided using *ReadCompleted* event. The process will run for a maximum of 5 seconds, or until success.

public void StartSearchAndReadStatus (int _maxScanTimeMs)

Parameter :

_maxScanTimeMs – maximum time in milliseconds that the function will try to find a TELID®.

Result :

-

Description :

Starts a *SearchAndReadStatus* process in a background thread. It searches for a TELID® logger near the antenna and reads its current logging status. This function completes immediately after being call, and results are provided using *ReadCompleted* event. The process will run for a maximum of milliseconds provided by parameter, or until success.

Read log functions

This chapter describes functions available for reading the measurements saved in TELID® data logger memory.

public TelidStateInformation ReadLogProtocol (TelidStateInformation _telidStatus)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to read.

Result :
success – returns instance of *TelidStateInformation* representing information, logging status and measurements of TELID® logger.
error – returns *null*.

Description :
Reads the log protocol of the desired TELID® logger and returns the instance of *TelidStateInformation* updated with all the available measurements saved in memory. This function blocks the execution until the answer is received or if a maximum of 20 times the read process failed internally.

public TelidStateInformation ReadLogProtocol (TelidStateInformation _telidStatus, int _maxReadErrors)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to read.
_maxReadErrors – maximum number of times to try the read process.

Result :
success – returns instance of *TelidStateInformation* representing information, logging status and measurements of TELID® logger.
error – returns *null*.

Description :
Reads the log protocol of the desired TELID® logger and returns the instance of *TelidStateInformation* updated with all the available measurements saved in memory. This function blocks the execution until the answer is received or if a maximum of read process failures provided by parameter.

public async Task<TelidStateInformation> ReadLogProtocolAsync (TelidStateInformation _telidStatus)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to read.

Result :
success – returns instance of *TelidStateInformation* representing information, logging status and measurements of TELID® logger.
error – returns *null*.

Description :
Reads the log protocol of the desired TELID® logger asynchronously and returns the instance of *TelidStateInformation* updated with all the available measurements saved in memory. This function blocks the execution until the answer is received or if a maximum of 20 times the read process failed internally.

public async Task<TelidStateInformation> ReadLogProtocolAsync (TelidStateInformation _telidStatus, int _maxReadErrors)

Parameter :

_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to read.
_maxReadErrors – maximum number of times to try the read process.

Result :

success – returns instance of *TelidStateInformation* representing information, logging status and measurements of TELID® logger.
error – returns *null*.

Description :

Reads the log protocol of the desired TELID® logger asynchronously and returns the instance of *TelidStateInformation* updated with all the available measurements saved in memory. This function blocks the execution until the answer is received or if a maximum of read process failures provided by parameter.

public void StartReadLogProtocol (TelidStateInformation _telidStatus)

Parameter :

_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to read.

Result :

-

Description :

Starts a *ReadLogProtocol* process in a background thread. It reads the log protocol of the desired TELID® logger asynchronously. This function completes immediately after being call, and results are provided using *ReadCompleted* event. The process will run for a maximum of 5 seconds, or until success.

public void StartReadLogProtocol (TelidStateInformation _telidStatus, int _ maxReadTimeMs)

Parameter :

_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to read.
_maxReadTimeMs – maximum time in milliseconds that the function will try to read the protocol of TELID®.

Result :

-

Description :

Starts a *ReadLogProtocol* process in a background thread. It reads the log protocol of the desired TELID® logger asynchronously. This function completes immediately after being call, and results are provided using *ReadCompleted* event. The process will run for a maximum of milliseconds provided by parameter, or until success.

Restart functions

This chapter describes functions available for starting a new log process of a TELID® data logger.
Warning: These functions will delete all previously saved measurements.

public bool RestartLog (LogStartParameters _startParams)

Parameter :
_startParams – instance of *LogStartParameters* class representing the parameters of the new measurement process.
Result :
success – returns *true*.
error – returns *null*.
Description :
Writes the given remarks and configures a new measurement in a TELID® logger near the antenna. This function blocks the execution until the answer is received or if a maximum of 3 times the restart process failed internally.

public bool RestartLog (LogStartParameters _startParams, int _maxProgramErrors)

Parameter :
_startParams – instance of *LogStartParameters* class representing the parameters of the new measurement process.
_maxReadErrors – maximum number of times to try the restart process.
Result :
success – returns *true*.
error – returns *null*.
Description :
Writes the given remarks and configures a new measurement in a TELID® logger near the antenna. This function blocks the execution until the answer is received or if a maximum of restart process failures provided by parameter.

public async Task<bool> RestartLogAsync (LogStartParameters _startParams)

Parameter :
_startParams – instance of *LogStartParameters* class representing the parameters of the new measurement process.
Result :
success – returns *true*.
error – returns *null*.
Description :
Writes the given remarks and configures a new measurement in a TELID® logger near the antenna asynchronously. This function blocks the execution until the answer is received or if a maximum of 3 times the restart process failed internally.

public async Task<bool> RestartLogAsync (LogStartParameters _startParams, int _maxProgramErrors)

Parameter :

_startParams – instance of *LogStartParameters* class representing the parameters of the new measurement process.
_maxReadErrors – maximum number of times to try the restart process.

Result :

success – returns *true*.
error – returns *null*.

Description :

Writes the given remarks and configures a new measurement in a TELID® logger near the antenna asynchronously. This function blocks the execution until the answer is received or if a maximum of restart process failures provided by parameter.

Stop functions

This chapter describes functions available for stopping current log process of a TELID® data logger.

public bool StopLog (TelidStateInformation _telidStatus)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to stop.
Result :
success – returns *true*.
error – returns *false*.
Description :
Stops the current log process of the desired TELID® logger. This function blocks the execution until the answer is received or if a maximum of 3 times the stop process failed internally.

public bool StopLog (TelidStateInformation _telidStatus, int _maxProgramErrors)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to stop.
_maxProgramErrors – maximum number of times to try the stop process.
Result :
success – returns *true*.
error – returns *false*.
Description :
Stops the current log process of the desired TELID® logger. This function blocks the execution until the answer is received or if a maximum of stop process failures provided by parameter.

public async Task<bool> StopLogAsync (TelidStateInformation _telidStatus)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to stop.
Result :
success – returns *true*.
error – returns *false*.
Description :
Stops asynchronously the current log process of the desired TELID® logger. This function blocks the execution until the answer is received or if a maximum of 3 times the stop process failed internally.

public async Task<bool> StopLogAsync (TelidStateInformation _telidStatus, int _maxProgramErrors)

Parameter :
_telidStatus – instance of *TelidStateInformation* containing the status of the TELID® data logger to stop.
_maxProgramErrors – maximum number of times to try the stop process.
Result :
success – returns *true*.
error – returns *false*.
Description :
Stops asynchronously the current log process of the desired TELID® logger. This function blocks the execution until the answer is received or if a maximum of stop process failures provided by parameter.

Classes, Enumerations, Events and Delegates

iIDReaderLibrary used utils classes

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

iIDReaderLibrary.Utils.ReaderIDInfo

This class contains information about the Reader.

Type	Property name	Description
int	ReaderID	Reader-ID (Serial number) of the RFID reader
string	HwInfo	String representation of the reader Hardware version
string	FwInfo	String representation of the running Firmware
string	HexString	Hexadecimal representation of class contents

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 *TelidControl* instance.

static InterfaceCommunicationSettings GetForUsbDevice ()

Parameter :
-
Result :
Instance of *InterfaceCommunicationSettings* class representing the parameters for USB communication
Description :
Initializes a new instance of *InterfaceCommunicationSettings* with default baud rate of 57600 to pass to *TelidControl* constructor.

static InterfaceCommunicationSettings GetForUsbDevice (int _baudrate)

Parameter :
_baudrate – baud rate value
Result :
Instance of *InterfaceCommunicationSettings* class representing the parameters for USB communication
Description :
Initializes a new instance of *InterfaceCommunicationSettings* to pass to *TelidControl* 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 *TelidControl* constructor.

static InterfaceCommunicationSettings GetForSerialDevice (int _portType, string _portName, int _baudrate)

Parameter :

_portType – number representing the port type. See *PortTypeEnum*

_portName – name of the serial port (not needed for USB port type)

_baudrate – baud rate value

Result :

Instance of *InterfaceCommunicationSettings* class representing the given parameters.

Description :

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

Utils classes

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

TELIDLibrary.Utils.TelidInformation

This class contains information of a TELID® data logger.

Type	Property name	Description
Int	SerialNumber	Serial number of the TELID® logger
TelidProducts	ProductType	Type of TELID® logger
PhysicalTypes[]	PhysicalData	Array of types of physical sensor types the logger is able to log

TELIDLibrary.Utils.TelidStateInformation

This class contains all the information contained in a TELID® data logger: type, status, measurements.

-	Property name	Description
TelidInformation	VersionInfo	Information of TELID® logger
String	FwVersion	Firmware version
String	Remarks	Measurement remarks
Bool	IsLogging	True if logger currently logging measurements
DateTime	ProgrammedTime	Time the new process was programmed (UTC)
DateTime	StartTime	Time the measurement started (UTC)
DateTime	StopTime	Time the measurement should stop (UTC) Note: If < 2020, no StopTime configured. The logger will log until memory is full
Int	NumberDatasetsLogged	Number of datasets saved in logger memory
Double	MemoryPercentageUsed	Percentage of memory used
ISensorParam	SensorParameters	Instance of class implementing *ISensorParam* interface. This class represents the sensor specific parameters
TelidMeasurement[]	LoggedMeasurements	Array of logged measurements. Hint: Is *null* until the protocol is read

TELIDLibrary.Utils.TelidMeasurement

This class contains sensor measurement details.

Type	Property name	Description
DateTime	Timestamp	Timestamp when the measurement took place
SensorValue[]	Values	Array of measured sensor values
Int	BatState	Battery state (not implemented)
Int	Status	Status byte (not implemented)
Int	ReaderID	Serial number of reader used to read the measurement (not implemented)

TELIDLibrary.Utils.SensorValue

This class contains individual sensor value details.

Type	Property name	Description
double	Magnitude	Magnitude of measured sensor value
string	Unit	Unit representing the sensor value unit
PhysicalTypes	ValueType	Enum value representing the sensor value type

TELIDLibrary.Utils.LogStartParameters

This class contains parameters to be used for a Restart process.

Type	Property name	Description
DateTime	StartTime	Time the log process should start
DateTime	StopTime	Time the log process should stop (only supported by some TELID® data loggers)
ISensorParam	SensorParams	Instance of *ISensorParam* representing the sensor parameters
TelidStateInformation	TelidStatus	Previously read TELID® log status

To initialize the class, the following static functions are provided.

LogStartParameters Initialize_TELID300nfc (TelidStateInformation _telidStatus, DateTime _startTime, DateTime _stopTime, TELID300nfc_SensorParam _sensorParams)

Parameter :

_telidStatus – previously read TELID® information and log status.
_startTime – Time the log process should start.
_stopTime – Time the log process should stop. Tipp: To force the TELID® logger to log measurements until the memory is full use a Time well in the future (for example 2099-12-31)
_sensorParams – instance of *TELID300nfc_SensorParam* that represents the sensor parameters.

Result :

Instance of *LogStartParameters* class representing the given parameters.

Description :

Initializes a new instance of *LogStartParameters* to start a new log process.

LogStartParameters Initialize_TELID300v1 (TelidStateInformation _telidStatus, DateTime _startTime, TELID311_SensorParam _sensorParams)

Parameter :

_telidStatus – previously read TELID® information and log status.
_startTime – Time the log process should start.
_sensorParams – instance of *TELID311_SensorParam* that represents the sensor parameters.

Result :

Instance of *LogStartParameters* class representing the given parameters.

Description :

Initializes a new instance of *LogStartParameters* to start a new log process.

LogStartParameters Initialize_TELID300v2 (TelidStateInformation _telidStatus, DateTime _startTime, DateTime _stopTime, ISensorParam _sensorParams)

Parameter :

_telidStatus – previously read TELID® information and log status.
_startTime – Time the log process should start.
_stopTime – Time the log process should stop. Tipp: To force the TELID® logger to log measurements until the memory is full use a Time well in the future (for example 2099-12-31)
_sensorParams – instance of class that implements *ISensorParam* that represents the sensor parameters

Result :

Instance of *LogStartParameters* class representing the given parameters.

Description :

Initializes a new instance of *LogStartParameters* to start a new log process.

Enumerations

This chapter describes enumerations that can be used in various functions described above.

iIDReaderLibrary.Utils.Definitions.PortTypeEnum

This enumeration contains values for the supported protocol type values.

Name	Value	Description
PortType_Serial	0	RS232 or USB in VCP mode
PortType_Bluetooth	2	Bluetooth™ SPP
PortType_USB	4	USB interface (no VCP mode)

TELIDLibrary.Utils.Definitions.PhysicalTypes

This enumeration contains different possible physical sensor types.

Name	Description
Temperature	Temperature in °C
RelativeHumidity	Relative humidity in %
Pressure	Pressure in mBar

TELIDLibrary.Utils.Definitions.TelidProducts

This enumeration contains the supported TELID® loggers.

Name	Description
TELID312_nfc	TELID® 312.nfc
TELID311	TELID® 311

Delegates

This chapter describes delegates that define events used in this *TelidControl* 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 ReadCompletedEventHandler (object _sender, TelidStateInformation _telidStateInfo)

Parameter :
_sender – object instance where the event handler is attached.
_telidStateInfo – instance of read status, or *null* if read process failed.
Result :
-
Description :
Represents a method that will handle the result to a read process.

public delegate void ReadLogProtocol_ProgressPercentageEventHandler (object _sender, int _percentage, TimeSpan _remaining)

Parameter :
_sender – object instance where the event handler is attached.
_percentage – percentage of log protocol read.
_remaining – approximate time remaining to complete.
Result :
-
Description :
Represents a method that will handle the update of *ReadLogProtocol* progress.

Events

This chapter describes events defined in *TelidControl* class.

public event InitializeCompletedEventHandler InitializeCompleted

Description :

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

public event ReadCompletedEventHandler ReadCompleted

Description :

This event is raised when any read operation running in separate background thread has a result to report.

public event ReadLogProtocol_ProgressPercentageEventHandler ReadLogProtocol_ProgressPercentageUpdated

Description :

This event is raised when reading log protocol to notify the current percentage progress and the approximate remaining time to complete.

Library configuration

This chapter describes parameters that may be changed to perform/change some processes in the library.

TELIDLibrary.Configuration.TELIDnfc_Parameters

This class contains parameters for handling TELID300.nfc loggers.

Type	Property name	Description
byte[]	AccessPassword	Password used for password protected TELID300.nfc

Appendix

Error - Codetable

State	Error
0x00	no error
RFID interface error codes	
0x23	TAG-error
0x24	no TAG near antenna
0x26	OP-CODE unknown
0x28	protocol failure
0x29	unknown TAG instruction
0x2A	unknown TAG-error
0x2B	error writing to TAG
0x2C	error reading from TAG
0x2E	error control-reading TAG
0x2F	wrong data control-reading TAG (RAW)
0x30	error in CRC-CheckSum
others	Other hardware specific error codes may occur – see hardware documentation.
Driver error codes	
0x08	identifiers do not match
0x11	range error
0x14	General TAG-error
0x3F	Communication or port error
0x43	Configuration error
0x4F	unknown/not supported option detected
0xFF	general communication or driver error

See hardware documentation for further error codes.