API Documentation

Product / Project:	**iID® MPC .Net class library**
Customer / Project Code:	**MPC enabled iID® contactless products**
Product:	iID® POCKETmini iID® POCKETwork
Product Code:	**-**
Revision:	**1.2.0.0**
Date:	**2014-09-30**

API - Definition for Windows® mobile and PC Windows®

This documents describes the .Net software API for communication with MPC enabled iID® contactless interfaces for MICROSOFT Windows® based PC systems.
iID® MPC class library is a .Net class for comfortable data loading from the memory of iID® contactless interfaces.

Class library name:	**microsensysDevTools.dll** **microsensysMPC.dll**
Name space:	**microsensys.MPC**
Tested devices:	Windows® 7 x64, Windows® Vista x86
Charset:	Windows® ANSI charset

Content

1. Short history

This chapter includes a short history of modifications of iID® MPC Library.

Date	Reason	Modification	Release Date / Version	FileName
2014-01-30	- APIDoc: base document		1.0	microsensysMPC.dll
2014-03-	- APIDoc: new version based on MPC library 1.0.1.0	Optimized interface detection, bug fix in Dispose	1.0.1.0	microsensysMPC.dll
2014-08-29	- APIDoc: new version based on MPC library 1.1.0.0	Implementation of battery state calculation New base communication class: - faster communication - multi protocol support	1.1.0.0	microsensysMPC.dll
2014-09-11	- APIDoc: new version based on MPC library 1.2.0.0	- improved data transfer including MPC memory error correction	1.2.0.0	microsensysMPC.dll

2. Introduction

iID® MPC library is used to load the data from any MPC enabled iID® contactless device. This implementation continues the native implementation within iID® driver engine.
Microsensys provides a sample code “MPCLibraryTest” showing the usage of this library.
Following external resources are required to compile your project:

`microsensysDevTools.dll`

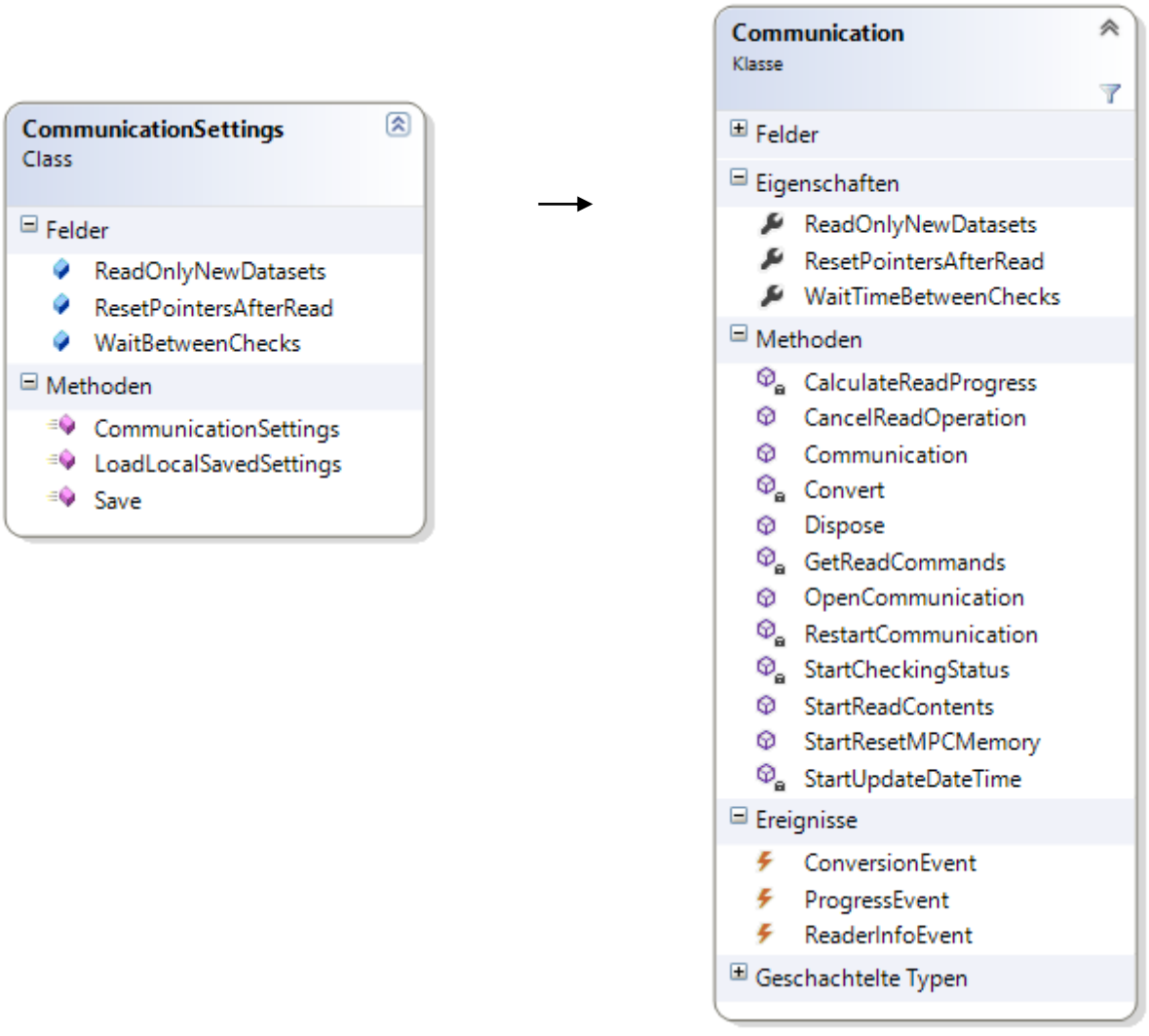
3. Class structure and main functionality

This chapter shows the library's class structure and it's main funtionality.

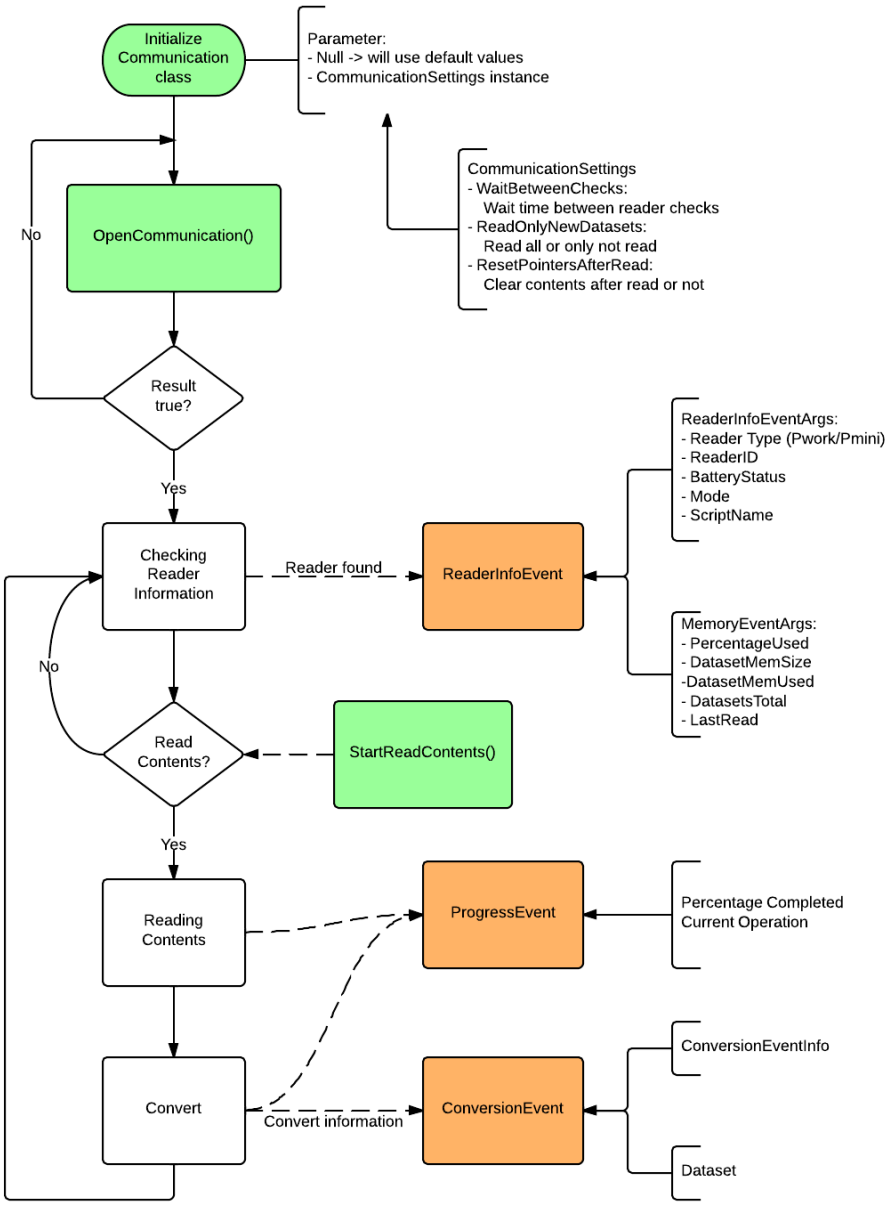
Prior using this library communication related settings have to be stored using microsensys reader connection tool.
The application software has to create an instance of microsensys.MPC.Communication
(the reader communication and data interface) as well as three event handlers for reader information, progress and data conversion:

```
microsensys.MPC.Communication.ReaderInformationEventHandler
microsensys.MPC.Communication.ConversionEventHandler
microsensys.MPC.Communication.ProgressEventHandler
```

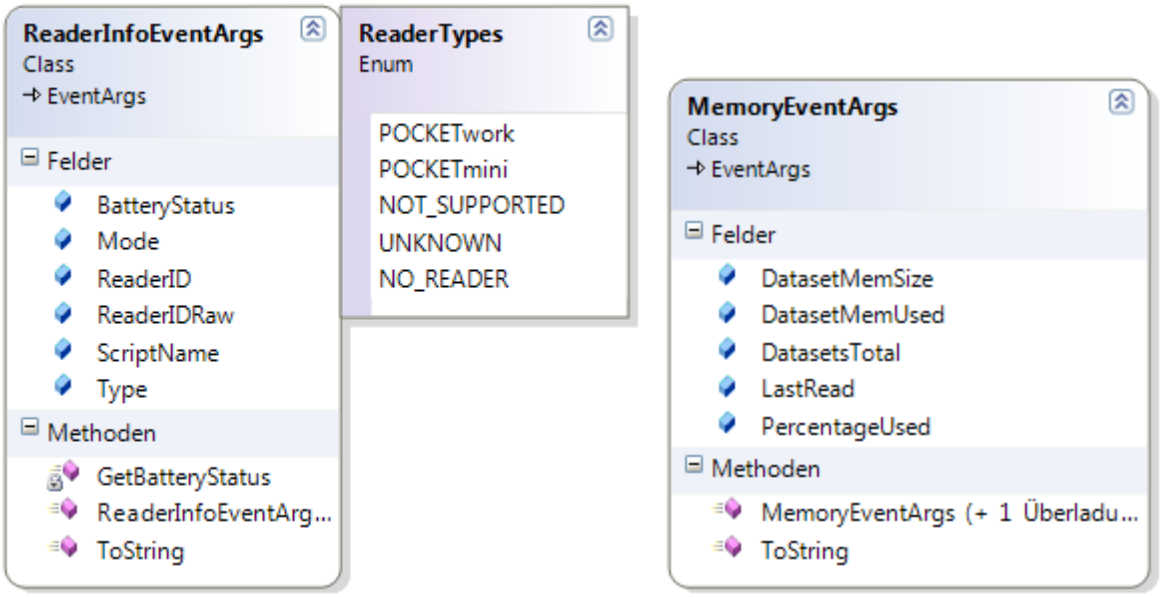
There are some properties, which can be set prior reading data from the MPC enabled device:



4. Flow Diagram



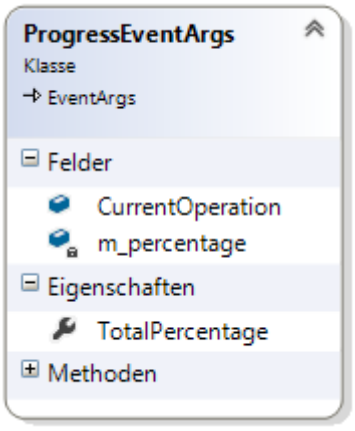
When a reader connection is established, there are to events thrown to the application, one containing reader related information `microsensys.MPC.Events.ReaderInfoEventArgs`, the second `microsensys.MPC.Events.MemoryEventArgs` having information about the device memory usage.



When getting this information, the data load process can be started using `StartReadingContents()`.

The running process again throws two events `ConversionEvent` and `ProgressEvent` containing the data read as well as information about the loading progress:

`microsensys.MPC.Events.ConversionEventInfo`
`microsensys.MPC.Events.ProgressEventArgs`



`CancelReadOperation()` can be used to stop the reading process.

4. Interface Description

4.1. microsensys.MPC.Communication

```
microsensys.MPC.ICommunication
```

This class is the main class establishing reader communication and providing functionality to load data from the device.

4.1.1. Properties

bool ReadOnlyNewDatasets { get; set; }

This property is used to load only new data from the device (true) or read the whole data content (false).

bool ResetPointersAfterRead { get; set; }

This property is used to reset the device data after successful reading (recommended). Therefor ResetPointersAfterRead has to be set true.

int WaitTimeBetweenChecks { get; set; }

This property is internal to change the reader communication timing.

4.1.2. Methods

bool OpenCommunication();

This methods opens the communication interface and tries to get data about connected reader and the data storage. Therefor the following event handler is used to transfer the data into the application:

```
event Communication.ReaderInformationEventHandler ReaderInfoEvent;
```

There are two structures used, one containg the reader related information, one containing information about the memory storage.

```
public class ReaderInfoEventArgs : EventArgs, microsensys.MPC.Events.ReaderInfoEventArgs
{
	public ReaderTypes Type;
	public int ReaderID;
	public byte[] ReaderIDRaw;
	public string BatteryStatus;
	public string Mode;
	public string ScriptName;
}
```

```
public class MemoryEventArgs : EventArgs
{
	public int PercentageUsed;
	public int DatasetMemSize;
	public int DatasetMemUsed;
	public int DatasetsTotal;
	public DateTime LastRead;
}
```

void CancelReadOperation();

CancelOperation() can be used to abort the current load operation.

void StartReadContents();

Using this method, the loading process can be started. During the load process there are two events thrown, telling the application about the progress and data:

```
event Communication.ProgressEventHandler ProgressEvent;
public class ProgressEventArgs : EventArgs
{
	private int m_percentage;
	public int TotalPercentage
	{
		get { return m_percentage; }
		set
		{
			if (value > 100) m_percentage = 100;
			else if (value < 0) m_percentage = 0;
			else m_percentage = value;
		}
}
	public string CurrentOperation;
}

event Communication.ConversionEventHandler ConversionEvent;
public enum ConversionEventInfo
{
	START,
	NewDataset,
	COMPLETED,
	FAILED,
	NO_DATA
}
```

void StartResetMPCMemory();

This function can be used to initialize MPC memory of your device. `StartResetMPCMemory()` will clear available datasets from your MPC device.

void StartUpdateDateTime(bool _isPOCKETwork);

Using this function, date and time of your PC will be deployed to your MPC device for time synchronisation. `_isPOCKETwork` has to be set „true“ in case of iID® POCKETwork hardware.

void Dispose();

Dispose frees the resources used by this library.

5. Error - Codetable

State	*Error*
0x00..0x30	See reader hardware documentation
Additional Errors	
0x3F	Communication or port error
0x4F	unknown/not supported option detected
0xFF	general communication or driver error