4.4.11 setProgressbar (only for PKT 4000) .......................................................... 36
4.4.12 setRFID (only for PKT 4000) ...................................................................... 36
4.4.13 Product_Version ......................................................................................... 37
4.4.14 Load_Scanner_Default (only for PKT 4000) ................................................. 37
4.4.15 EnableScanning (only for PKT 4000) ......................................................... 37
4.4.16 ReadRFIDwithMRZ (only for PKT 4000) .................................................... 38
4.4.17 ReadRFIDwithCAN (only for PKT 4000, not yet supported) ..................... 38
4.4.18 EnableOCR .............................................................................................. 38
4.4.19 getEnableScanner (only for PKT 4000) ....................................................... 39
4.4.20 setEnableScanner (only for PKT 4000) ...................................................... 39
4.4.21 getEnableOCR .......................................................................................... 39
4.4.22 setEnableOCR .......................................................................................... 39
4.4.23 getEnableMSR .......................................................................................... 40
4.4.24 setEnableMSR .......................................................................................... 40
4.4.25 getEnableRFID (only for PKT 4000) ......................................................... 40
4.4.26 setEnableRFID (only for PKT 4000) ......................................................... 40

4.4 DeviceLED (only for PKT 4000) ..................................................................... 41

4.5 DeviceBuzzer .................................................................................................. 42

4.6 PKT 4000 Exception ....................................................................................... 43
Disclaimer

PrehKeyTec GmbH reserves the right to make changes in specifications and other information contained in this document without prior notice. The reader should consult PrehKeyTec whether any such changes have been made. The information in this manual does not represent a commitment on the part of PrehKeyTec.

Whilst every care has been taken in producing this manual PrehKeyTec shall not be liable for technical or editorial errors or omissions contained herein, nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

This document contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be copied without the prior written consent of PrehKeyTec.

© 2014 PrehKeyTec GmbH. All rights reserved.

Microsoft® Windows® is a trademark of Microsoft Corporation.

Other product names or marks mentioned in this document may be trademarks or registered trademarks of other companies and are the property of their respective owners.
## History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>05.11.2014</td>
<td>Document created</td>
</tr>
<tr>
<td>1.2</td>
<td>23.01.2015</td>
<td>PKT4000 Methods need not be allocated with “new”. DataEvents are separated. Bar coded boarding passes according IATA Resolution 792 in BarcodeDataEvent added. Status Codes added. Buzzer command added.</td>
</tr>
<tr>
<td>1.3</td>
<td>31.07.2015</td>
<td>Scanner Enable and Disable method. Some additional ERROR CODES.</td>
</tr>
<tr>
<td>1.4</td>
<td>11.02.2016</td>
<td>Older French ID Card added. Data input reworked. RFID image can be saved. Read MRZ and image from RFID chip with document number, date of birth and date of expiry.</td>
</tr>
<tr>
<td>1.4.3.0</td>
<td>11.04.2016</td>
<td>Read MRZ and image from RFID chip with CAN (Card Access Number) Additional parameter (PassPortStatus) in RFIDData Event added.</td>
</tr>
<tr>
<td>1.5.0.0</td>
<td>07.12.2016</td>
<td>MCI 111 A and ML 2 A added.</td>
</tr>
<tr>
<td>1.6.0.0</td>
<td>15.02.2017</td>
<td>Enable/disable OCR, MSR, RFID Possible dead lock in event present/remove document. No reread delay in EnableScanning.</td>
</tr>
</tbody>
</table>
1. General

1.1. PKT 4000
The PKT 4000 is a document scanner and imager ideally suited for reading the following document types:

- Passports
- ID cards
- Visa
- ePassports
- Barcodes

The device has been designed to read both 1D and 2D barcodes and the MRZ (Machine Readable Zone) of travel documents, typically Passports. An additional feature of the PKT 4000 is the ability to capture pictures from documents, again typically being Passport images and photographs. An optional RFID reader communicates with the embedded RFID chip within ePassports, providing the capture of ePassport data along with digital image information such as facial profiles etc. The connection of the PKT 4000 is via USB interface, noting that device drivers are required in order to work the various PKT 4000 device modules. Our recommendation would be that you install all the device drivers before working with the SDK and the PKT 4000.

1.2. MCI 111 A / ML 2 A
The keyboard MCI 111 A and the stand alone module ML 2 A is ideal for reading the following documents which must be compliant to ICAO 9303:

- Passports
- ID cards
- Visa

The MCI 111 A and the ML A are equipped with an OCR reader to scan the MRZ of optical readable travel documents. The MSR (Magnetic Stripe Reader) reads up to three tracks of documents with magnetic stripe. The user can swipe the documents from both directions. The devices supports USB interface and don’t need an additional external power supply. Our recommendation would be that you install all the device drivers before working with the SDK and the MCI 111 A / ML 2 A.
1.3. ML 4
The ML 4 is a RFID reader which can read the MRZ (Data Group 1) and picture of the face (Data Group 2) of ePassports and eID documents according the BAC protocol. This standalone device is the optimal combination with the MCI 111 A / ML 2 A to upgrade the functionality of reading RFID passports and ID cards.

Note:
If you want to use the SDK with ML 4 and MCI 111 A / ML 2 A, please see the functions of PKT 4000.

2. Requirements
Currently the only specific requirements are as follows, although please feel free to check with Technical support if you have any additional requirements of questions on this point:

Windows® XP
Windows® 7 (32 bit / 64 bit)
Windows® 10 (32 bit / 64 bit)
.NET Framework 4

3. PKT4000DLL
The interface between the user application and the PKT 4000 / MCI 111 A / ML 2 A / ML 4 is the PKT4000DLL. This library is written in C# with .NET framework 4. All required ports and settings are configured from within the interface. The following chapters describe the methods and parameters of the PKT4000DLL and how to implement them as user source code.
4. SDK Description

4.1 Implementation
Start by adding the PKT4000DLL into your application, the following data should be seen in the properties of your application:

```csharp
using PKT4000DLL;
```

Select to enable you to have access to the features, methods and parameters of the PKT4000DLL.

4.2 Raw Data Protocol
The following control characters are used to control the data flow from the PKT 4000 to the host.

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>HEX VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>02</td>
<td>START OF TEXT. This character signals the START of a BLOCK.</td>
</tr>
<tr>
<td>ETX</td>
<td>03</td>
<td>END OF TEXT. This character signals the END of a BLOCK.</td>
</tr>
<tr>
<td>SOD1</td>
<td>OE</td>
<td>Start delimiter for Magnetic Swipe Reader (MSR).</td>
</tr>
<tr>
<td>EOD1</td>
<td>OF</td>
<td>End delimiter for Magnetic Swipe Reader (MSR).</td>
</tr>
<tr>
<td>SOD2</td>
<td>1C</td>
<td>Start delimiter for OCR devices.</td>
</tr>
<tr>
<td>EOD2</td>
<td>1D</td>
<td>End delimiter for OCR devices.</td>
</tr>
<tr>
<td>SOD3</td>
<td>1E</td>
<td>Start delimiter for Barcode Reader.</td>
</tr>
<tr>
<td>EOD3</td>
<td>1F</td>
<td>End delimiter for Barcode Reader.</td>
</tr>
</tbody>
</table>

A BLOCK has the following structure:

```
<SODn><STX>...data...<ETX><EODn>
```
4.3 Connecting to the device

4.3.1 Event Handler

There are different Event Handlers which receive the data from the Modules. The Event Handler Delegate represents the method that will handle an event.

4.3.2 Event Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BarcodeEvent</td>
<td>Event to receive barcode data</td>
</tr>
<tr>
<td>MSRDataEvent</td>
<td>Event to receive magnetic stripe data (MSR)</td>
</tr>
<tr>
<td>OCRDataEvent</td>
<td>Event to receive OCR data from travel documents</td>
</tr>
<tr>
<td>RFIDDataEvent</td>
<td>Event to receive RFID data (e. g. digital stored picture) of ePassports</td>
</tr>
<tr>
<td>ScanDataEvent</td>
<td>Event to receive a picture of the scanned document</td>
</tr>
<tr>
<td>StatusMessageEvent</td>
<td>Event to receive status message data and status code</td>
</tr>
<tr>
<td>ProgressbarEvent</td>
<td>Event to receive progress of read process of RFID data</td>
</tr>
<tr>
<td>PKT4000VersionEvent</td>
<td>Event to receive version PKT4000 device</td>
</tr>
</tbody>
</table>
4.3.3 Create an Event Object

Create an object of the class `OnDataEvent`.

Example:

```csharp
OnDataEvent events = new OnDataEvent();
```

4.3.4 Add Events

The following events are defined in the new object:

4.3.4.1 Barcode Data Event

Create and add the event with the plus “+=” operator to invoke the method.

Example:

```csharp
events.BarcodeEvent += new EventHandler<BarcodeDataEvent>(this.yourBarCodeData);
```

Definition of the method assigned to the EventHandler delegate.

```csharp
private void yourBarCodeData(object sender, BarcodeDataEvent e) {} 
```

If the BarcodeEvent is triggered, the defined method `yourBarCodeData` is called. The received data is divided into the variables below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string barcode</code></td>
<td>Contains received barcode data as a string.</td>
</tr>
<tr>
<td><code>string barcodetype</code></td>
<td>Contains the type of the barcode.</td>
</tr>
<tr>
<td><code>byte[] barcode_data</code></td>
<td>Contains raw data of barcode.</td>
</tr>
<tr>
<td><code>BarcodeData_S barcode_data_s</code></td>
<td>See table below. Only for bar coded boarding pass according IATA Resolution 792.</td>
</tr>
<tr>
<td><code>BarcodeData_M barcode_data_m</code></td>
<td>See table below. Only for bar coded boarding pass according IATA Resolution 792.</td>
</tr>
</tbody>
</table>
Barcode types:
- Code Interleave 2 of 5 - QR
- Code Industrial 2 of 5 - Aztec
- Code 39 - EAN13
- Data Matrix - PDF417
- Code 128

Additional barcode types on request.

Raw Barcode data:

<SOD3><STX>I Bar Code Data <CR><ETX><EOD3>

Where: I = Bar Code ID;

‘1’ (0x31) = Code Interleaved 2 of 5
‘2’ (0x32) = Code Industrial 2 of 5
‘3’ (0x33) = Code 39
‘4’ (0x34) = Data Matrix
‘5’ (0x35) = Code 128
‘6’ (0x36) = PDF417
‘7’ (0x37) = QR
‘8’ (0x38) = Aztec
‘A’ (0x41) = EAN13

Bar Coded Boarding Pass Format Code S:

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Element Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>barcode_data_s.item_no_01</td>
<td>Format Code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_11</td>
<td>Passenger Name</td>
</tr>
<tr>
<td>barcode_data_s.item_no_13</td>
<td>Operating carrier’s PNR code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_15</td>
<td>Servicing Carrier/System Provider controlling PNR</td>
</tr>
<tr>
<td>barcode_data_s.item_no_17</td>
<td>Passenger description</td>
</tr>
<tr>
<td>barcode_data_s.item_no_21</td>
<td>Date of issue of the document accepted in exchange for this boarding pass (Julian date)</td>
</tr>
<tr>
<td>barcode_data_s.item_no_22</td>
<td>Date of issue (Julian Date)</td>
</tr>
</tbody>
</table>

Copyright PrehKeyTec © 2014 – 2017
<table>
<thead>
<tr>
<th>Barcode Data Item No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>barcode_data_s.item_no_25</td>
<td>X/O Indicator</td>
</tr>
<tr>
<td>barcode_data_s.item_no_26</td>
<td>From City Airport Code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_32</td>
<td>Airline/agent code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_38</td>
<td>To City Airport Code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_42</td>
<td>Airline Designator</td>
</tr>
<tr>
<td>barcode_data_s.item_no_43</td>
<td>Flight Number</td>
</tr>
<tr>
<td>barcode_data_s.item_no_46</td>
<td>Date of Flight (Julian Date)</td>
</tr>
<tr>
<td>barcode_data_s.item_no_71</td>
<td>Compartment Code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_84</td>
<td>Airport/passenger processing indicator</td>
</tr>
<tr>
<td>barcode_data_s.item_no_89</td>
<td>ID/AD indicator</td>
</tr>
<tr>
<td>barcode_data_s.item_no_104</td>
<td>Seat Number</td>
</tr>
<tr>
<td>barcode_data_s.item_no_107</td>
<td>Check-In Sequence Number</td>
</tr>
<tr>
<td>barcode_data_s.item_no_108</td>
<td>International documentation verification</td>
</tr>
<tr>
<td>barcode_data_s.item_no_113</td>
<td>Passenger Status</td>
</tr>
<tr>
<td>barcode_data_s.item_no_118</td>
<td>Free baggage allowance</td>
</tr>
<tr>
<td>barcode_data_s.item_no_1</td>
<td>Coupon number of document accepted in exchange for this boarding pass</td>
</tr>
<tr>
<td>barcode_data_s.item_no_142</td>
<td>Airline numeric code</td>
</tr>
<tr>
<td>barcode_data_s.item_no_2</td>
<td>Document form/derail number of the document accepted in exchange for this boarding pass</td>
</tr>
<tr>
<td>barcode_data_s.item_no_3</td>
<td>Document number check digit of the document accepted in exchange for this boarding pass</td>
</tr>
<tr>
<td>barcode_data_s.item_no_198</td>
<td>Origin</td>
</tr>
<tr>
<td>barcode_data_s.item_no_199</td>
<td>Destination</td>
</tr>
<tr>
<td>barcode_data_s.item_no_236</td>
<td>Frequent flyer number</td>
</tr>
<tr>
<td>barcode_data_s.item_no_240</td>
<td>Change-in gauge seat number or additional seat information</td>
</tr>
<tr>
<td>barcode_data_s.item_no_253</td>
<td>Electronic Ticket Indicator</td>
</tr>
<tr>
<td>barcode_data_s.item_no_4</td>
<td>For Individual Airline Use</td>
</tr>
</tbody>
</table>
Bar Coded Boarding Pass Format Code M:

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Element Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>barcode_data_m.item_no_01</td>
<td>Format Code</td>
</tr>
<tr>
<td>barcode_data_m.item_no_05</td>
<td>Number of Segments Encoded</td>
</tr>
<tr>
<td>barcode_data_m.item_no_11</td>
<td>Passenger Name</td>
</tr>
<tr>
<td>barcode_data_m.item_no_253</td>
<td>Electronic Ticket Indicator</td>
</tr>
<tr>
<td>barcode_data_m.item_no_07</td>
<td>Operating carrier’s PNR code</td>
</tr>
<tr>
<td>barcode_data_m.item_no_26</td>
<td>From City Airport Code</td>
</tr>
<tr>
<td>barcode_data_m.item_no_38</td>
<td>To City Airport Code</td>
</tr>
<tr>
<td>barcode_data_m.item_no_42</td>
<td>Airline Designator</td>
</tr>
<tr>
<td>barcode_data_m.item_no_43</td>
<td>Flight Number</td>
</tr>
<tr>
<td>barcode_data_m.item_no_46</td>
<td>Date of Flight (Julian Date)</td>
</tr>
<tr>
<td>barcode_data_m.item_no_71</td>
<td>Compartment Code</td>
</tr>
<tr>
<td>barcode_data_m.item_no_104</td>
<td>Seat Number</td>
</tr>
<tr>
<td>barcode_data_m.item_no_107</td>
<td>Check-In Sequence Number</td>
</tr>
<tr>
<td>barcode_data_m.item_no_113</td>
<td>Passenger Status</td>
</tr>
<tr>
<td>barcode_data_m.item_no_6</td>
<td>Field size of following “For individual airline use” field</td>
</tr>
<tr>
<td>barcode_data_m.item_no_4</td>
<td>For Individual Airline Use</td>
</tr>
</tbody>
</table>

Copyright PrehKeyTec © 2014 – 2017
Example:

delegate void BarCodehelper(object sender, BarcodeDataEvent e);
private void yourBarCodeData(object sender, BarcodeDataEvent e)
{
    if (InvokeRequired)
    {
        BarCodehelper barcode = new BarCodehelper(yourBarCodeData);
        this.Invoke(barcode, sender, e);
    }
    else
    {
        string Barcode = e.barcode;
        string Barcodetype = e.barcodetype;
        // boarding pass detected
        if (e.barcode_data_m.item_no_01 == "M")
        {
            // boarding pass with format code M
            // ...
        }
    }
}
4.3.4.2 Magnetic Swipe Reader (MSR) Data Event

Create and add the event with the plus “+=” operator to invoke the method.

Example:
```
events.MSRDataEvent += new EventHandler<MSRDataEvent>(this.yourMSRData);
```

Definition of the method assigned to the EventHandler delegate.
```
private void yourMSRData(object sender, MSRDataEvent e){}
```

If the MSRDataEvent is triggered, the defined method yourMSRData is called.
The received data is divided into the four variables.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte[] bData</td>
<td>Contains all received raw data.</td>
</tr>
<tr>
<td>string MSRdatastring1</td>
<td>Contains the data of the first line.</td>
</tr>
<tr>
<td>string MSRdatastring2</td>
<td>Contains the data of the second line.</td>
</tr>
<tr>
<td>string MSRdatastring3</td>
<td>Contains the data of the third line.</td>
</tr>
</tbody>
</table>

Raw MSR data:

```
<SOD1><STX><DOC-ID><T/B>magntic data
  <CR><T/B>magntic data
  <CR><T/B>magntic data
  <CR><T/B>magntic data
  <CR><T/B>magntic data
  <CR><T/B>magntic data<ETX>
```
Where:

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>ASCII</th>
<th>HEX</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC-ID</td>
<td>A or C</td>
<td>41 or</td>
<td>An ATB document read. A credit card read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>T/B</td>
<td>11 to 13, 21 to 23, 31 to 33, 41 to 43</td>
<td>TRACK/BLOCK NUMBER. High order nibble = Track Number. Low order nibble = Block Number</td>
<td></td>
</tr>
</tbody>
</table>

Example:

```csharp
delegate void MSRhelper(object sender, MSRDataEvent e);
private void yourMSRData(object sender, MSRDataEvent e)
{
    if (InvokeRequired)
    {
        MSRhelper msr = new MSRhelper(yourMSRData);
        this.Invoke(msr, sender, e);
    }
    else
    {
        byte[] msr = e.bData;
        string msr_line1 = e.MSRdatastring1;
        string msr_line2 = e.MSRdatastring2;
        string msr_line3 = e.MSRdatastring3;
    }
}
```
4.3.4.3 Optical Character Recognition (OCR) Data Event

Create and add the event with the plus “+=” operator to invoke the method.

Example:

```csharp
events.OCRDataEvent += new Event<OCRDataEvent>(this.yourOCRData);
```

Definition of the method assigned to the Event<Delegate> delegate.

```csharp
private void yourOCRData(object sender, OCRDataEvent e){ }
```

If the OCRDataEvent is triggered, the defined method yourOCRData is called.

The received data is divided into the following variables.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte[] ocr_data</td>
<td>Contains all received raw data.</td>
</tr>
<tr>
<td>string type</td>
<td>Contains the type of the passport.</td>
</tr>
<tr>
<td>string code</td>
<td>Contains the code of the passport.</td>
</tr>
<tr>
<td>string passportnr</td>
<td>Contains the passport number.</td>
</tr>
<tr>
<td>string surname</td>
<td>Contains the surname.</td>
</tr>
<tr>
<td>string givenname</td>
<td>Contains the given name.</td>
</tr>
<tr>
<td>string nationality</td>
<td>Contains the nationality.</td>
</tr>
<tr>
<td>string dateofbirth</td>
<td>Contains the date of birth.</td>
</tr>
<tr>
<td>string dateofexpiry</td>
<td>Contains the date of expiry.</td>
</tr>
<tr>
<td>string sex</td>
<td>Contains the sex.</td>
</tr>
<tr>
<td>string personalnumber</td>
<td>Contains the personal number.</td>
</tr>
<tr>
<td>string mrzline1</td>
<td>Contains the first MRZ (machine readable zone) line.</td>
</tr>
<tr>
<td>string mrzline2</td>
<td>Contains the second MRZ line.</td>
</tr>
<tr>
<td>string mrzline3</td>
<td>Contains the third MRZ line.</td>
</tr>
</tbody>
</table>
Example:

delegate void OCRhelper(object sender, OCRDataEvent e);
private void yourOCRData(object sender, OCRDataEvent e)
{
    if (InvokeRequired)
    {
        OCRhelper ocr = new OCRhelper(yourOCRData);
        this.Invoke(ocr, sender, e);
    }
    else
    {
        string ocrdata = e.ocrdata;
        string type = e.type;
        string code = e.code;
        string passportnr = e.passportnr;
        string surname = e.surname;
        string givenname = e.givenname;
        string nationality = e.nationality;
        string dateofbirth = e.dateofbirth;
        string dateofexpiry = e.dateofexpiry;
        string sex = e.sex;
        string personr = e.personalnumber;
        string mrzline1 = e.mrzline1;
        string mrzline2 = e.mrzline2;
        string mrzline3 = e.mrzline3;
    }
}
4.3.4.4 Radio-Frequency Identification (RFID) Data Event

Create and add the event with the plus “+=” operator to invoke the method.

Example:

```csharp
events.RFIDDataEvent += new EventHandler<RFIDDataEvent>(this.yourRFIDData);
```

Definition of the method assigned to the EventHandler delegate.

```csharp
private void yourRFIDData(object sender, RFIDDataEvent e){ }
```

If the RFIDDataEvent is triggered, the defined method yourRFIDData is called.

The received data is an image from the passport.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image RFIDImage</td>
<td>The digital stored image of the ePassport (DG2).</td>
</tr>
<tr>
<td>byte[] ocr_data</td>
<td>All received raw data in DG1.</td>
</tr>
<tr>
<td>string type</td>
<td>Type of the passport in DG1.</td>
</tr>
<tr>
<td>string code</td>
<td>Code of the passport in DG1.</td>
</tr>
<tr>
<td>string passportnr</td>
<td>Passport number in DG1.</td>
</tr>
<tr>
<td>string surname</td>
<td>Surname in DG1.</td>
</tr>
<tr>
<td>string givenname</td>
<td>Given name in DG1.</td>
</tr>
<tr>
<td>string nationality</td>
<td>Nationality in DG1.</td>
</tr>
<tr>
<td>string dateofbirth</td>
<td>Date of birth in DG1.</td>
</tr>
<tr>
<td>string sex</td>
<td>Gender of passport holder in DG1.</td>
</tr>
<tr>
<td>string dateofexpiry</td>
<td>Date of expiry in DG1.</td>
</tr>
<tr>
<td>string personalnumber</td>
<td>Personal number in DG1</td>
</tr>
<tr>
<td>string mrzline1</td>
<td>MRZ line 1 in DG1</td>
</tr>
<tr>
<td>string mrzline2</td>
<td>MRZ line 2 in DG1</td>
</tr>
<tr>
<td>string mrzline3</td>
<td>MRZ line 2 in DG1</td>
</tr>
</tbody>
</table>
PassPortStatus status

status.protocol[i].getName(), protocol name
status.protocol[i].getActive(), is protocol active or not
status.files[i].getName(), file name (e.g., “DG1”)
status.files[i].getActive(), file read successfully

This parameter is available from version 1.4.3.0

Example:

delegate void RFIDhelper(object sender, RFIDDataEvent e);
private void yourRFIDData(object sender, RFIDDataEvent e) {
  if (InvokeRequired)
  {
    RFIDhelper rfid = new RFIDhelper(yourRFIDData);
    this.Invoke(rfid, sender, e);
  }
  else
  {
    Image img = e.RFIDImage;
    // fill DG1 Text Box with DG1 data (MRZ of RFID chip)
    richTextBoxDG1.Text = e.mrzline1 + "\n" + e.mrzline2 + "\n" + e.mrzline3;
  }
}
### 4.3.4.5 Scanner Data Event

Create and add the event with the plus “+=” operator to invoke the method.

```csharp
events.ScanDataEvent += new EventHandler<ScanDataEvent>(this.yourScanData);
```

Definition of the method assigned to the EventHandler delegate.

```csharp
private void yourScanData(object sender, ScanDataEvent e){ }
```

The ScanDataEvent can be triggered in two different ways. The first way is a manual invocation of the CaptureImage method from the PKT4000Methods class.

```csharp
pktMethods.CaptureImage(events);
```

The second way is to set the boolean value of the setCaptureImage method from the PKT4000Methods class to `true`. If the OCRDataEvent is triggered, the CaptureImage method takes a picture from the document automatically.

```csharp
pktMethods.setCaptureImage(bool Capture);
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boolean</strong></td>
<td><strong>Capture</strong></td>
</tr>
<tr>
<td><code>true</code></td>
<td>= automatic capture on</td>
</tr>
<tr>
<td><code>false</code></td>
<td>= automatic capture off</td>
</tr>
</tbody>
</table>

Example:

```csharp
pktMethods.setCaptureImage(true);
```

With both methods, the event ScanDataEvent is called.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image</strong></td>
<td><strong>ScanImage</strong></td>
</tr>
<tr>
<td>Contains the image of the captured document</td>
<td></td>
</tr>
<tr>
<td><strong>Image</strong></td>
<td><strong>ScanFace</strong></td>
</tr>
<tr>
<td>Contains the detected face of the captured document if possible. Face detection must be switched on and is optional</td>
<td></td>
</tr>
<tr>
<td>pkt4000Methods.setFaceDetection(true);</td>
<td></td>
</tr>
</tbody>
</table>
4.3.4.6  Current State Data Event
Create and add the event with the plus “+=” operator to invoke the method.

Example:
```csharp
private void Capture_Image_Click(object sender, EventArgs e)
{
    pktMethods.CaptureImage(events);
}
delegate void Scanhelper(object sender, DataEvent e);
private void yourScanData(object sender, DataEvent e)
{
    if (InvokeRequired)
    {
        Scanhelper scan = new Scanhelper(yourScanData);
        this.Invoke(scan, sender, e);
    }
    else
    {
        Image img = e.ScanImage;
    }
}
```

**Definition of the method assigned to the EventHandler delegate.**

```csharp
private void yourStatusData(object sender, StatusDataEvent e){}
```

If the StatusMessageEvent is triggered, the defined method yourStatusData is called. The received data shows the current state message and the returned value from the method which is currently in use.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string Message</td>
<td>Status message</td>
</tr>
<tr>
<td>Error.ERROR_STATE</td>
<td>Status Code</td>
</tr>
</tbody>
</table>
## Status Codes:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td><code>Error.ERROR_STATE.SUCCESS</code></td>
<td>Successful</td>
</tr>
<tr>
<td>0x00000001</td>
<td><code>Error.ERROR_STATE.XML</code></td>
<td>XML file error</td>
</tr>
<tr>
<td>0x00000002</td>
<td><code>Error.ERROR_STATE.COM_PORT_SCANNER</code></td>
<td>COM port scanner error</td>
</tr>
<tr>
<td>0x00000003</td>
<td><code>Error.ERROR_STATE.COM_PORT_PKT4000</code></td>
<td>COM port PKT4000 error</td>
</tr>
<tr>
<td>0x00000004</td>
<td><code>Error.ERROR_STATE.COM_PORT_OFFLINE</code></td>
<td>Device offline</td>
</tr>
<tr>
<td>0x00000008</td>
<td><code>Error.ERROR_STATE.COM_PORT_ONLINE</code></td>
<td>Device online</td>
</tr>
<tr>
<td>0x00000010</td>
<td><code>Error.ERROR_STATE.VERSION</code></td>
<td>Read version error</td>
</tr>
<tr>
<td>0x00000011</td>
<td><code>Error.ERROR_STATE.VCOMP_SERVICE</code></td>
<td>VCOMP service error</td>
</tr>
<tr>
<td>0x00000012</td>
<td><code>Error.ERROR_STATE.VCOMS_SERVICE</code></td>
<td>VCOMS service error</td>
</tr>
<tr>
<td>0x00000020</td>
<td><code>Error.ERROR_STATE.READ_323</code></td>
<td>Read COM port error</td>
</tr>
<tr>
<td>0x00000040</td>
<td><code>Error.ERROR_STATE.MSR</code></td>
<td>MSR read error</td>
</tr>
<tr>
<td>0x00000080</td>
<td><code>Error.ERROR_STATE.READ_SCANNER</code></td>
<td>Read scanner error</td>
</tr>
<tr>
<td>0x00001000</td>
<td><code>Error.ERROR_STATE.READ_OCR</code></td>
<td>Read OCR data error</td>
</tr>
<tr>
<td>0x00002000</td>
<td><code>Error.ERROR_STATE.CHECK_PASSPORT_NUMBER</code></td>
<td>Check passport number error</td>
</tr>
<tr>
<td>0x0000201</td>
<td><code>Error.ERROR_STATE.CHECK_DATE_OF_BIRTH</code></td>
<td>Check date of birth error</td>
</tr>
<tr>
<td>0x0000202</td>
<td><code>Error.ERROR_STATE.CHECK_DATE_OF_EXPIRY</code></td>
<td>Check date of expiry error</td>
</tr>
<tr>
<td>0x0000204</td>
<td><code>Error.ERROR_STATE.CHECK_PERSONAL_NUMBER</code></td>
<td>Check personal number error</td>
</tr>
<tr>
<td>0x0000208</td>
<td><code>Error.ERROR_STATE.CHECK_COMPOSITE</code></td>
<td>Check overall digit error</td>
</tr>
<tr>
<td>0x0000220</td>
<td><code>Error.ERROR_STATE.FACE_DETECTION</code></td>
<td>Face detection error</td>
</tr>
<tr>
<td>0x0000300</td>
<td><code>Error.ERROR_STATE.RFID</code></td>
<td>RFID reader not available</td>
</tr>
<tr>
<td>0x0000301</td>
<td><code>Error.ERROR_STATE.NO_RFID_CHIP</code></td>
<td>No RFID chip available</td>
</tr>
<tr>
<td>0x0000302</td>
<td><code>Error.ERROR_STATE.SC_NOK</code></td>
<td>Smart Card Error</td>
</tr>
<tr>
<td>0x0000400</td>
<td><code>Error.ERROR_STATE.EVENT</code></td>
<td>Event error</td>
</tr>
</tbody>
</table>
4.3.4.7 Version Data Event

**Example:**

```csharp
delegate void StatusHelper(object sender, StatusDataEvent e);
private void yourStatusData(object sender, StatusDataEvent e)
{
   if (InvokeRequired)
   {
      StatusHelper stat = new StatusHelper(yourStatusData);
      this.Invoke(stat, sender, e);
   }
   else
   {
   }
}
```

Create and add the event with the plus "+=" operator to invoke the method.

**Example:**

```csharp
events.PKT4000VersionEvent += new EventHandler<VersionDataEvent>(this.yourPKT4000Version);
```

Definition of the method assigned to the EventHandler delegate.

```csharp
private void yourPKT4000Version(object sender, VersionDataEvent e){}
```

If the PKT4000VersionEvent is triggered, the defined method yourPKT4000Version is called. The received data shows the current version of the PKT4000 device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string copyright</td>
<td>Copyright information</td>
</tr>
<tr>
<td>string firmware</td>
<td>Version of firmware</td>
</tr>
<tr>
<td>string firmware_date_time</td>
<td>Date of firmware</td>
</tr>
<tr>
<td>string serial_number</td>
<td>Serial number of PKT 4000</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>string production_date</td>
<td>Date of production</td>
</tr>
<tr>
<td>string product_id</td>
<td>Product ID</td>
</tr>
<tr>
<td>string product_number</td>
<td>Product number</td>
</tr>
<tr>
<td>string dll_version</td>
<td>Version of PKT 4000 DLL</td>
</tr>
</tbody>
</table>

Example:

delegate void PKT4000_Versionhelper(object sender, VersionDataEvent e);
private void yourPKT4000Version(object sender, VersionDataEvent e)
{
    if (InvokeRequired)
    {
        PKT4000_Versionhelper pro = new PKT4000_Versionhelper (yourPKT4000Version);
        this.Invoke(pro, sender, e);
    }
    else
    {
        string copy = e.copyright;
    }
}
### 4.3.4.8 Progress Bar Data Event

The Progressbar is only useful for Devices with RFID reader to get the progress of reading out the digital stored image. The method `setProgressbar` must be set to true.

```
pktMethods.setProgressbar(true);
```

Create and add the event with the plus “+=” operator to invoke the method.

```
Example:
events.ProgressBarEvent += new EventHandler<ProgressbarDataEvent>(this.yourProgressbar);
```

Definition of the method assigned to the EventHandler delegate.

```
private void yourProgressbar(object sender, ProgressbarDataEvent e){}
```

The Progressbar is available if a RFID event is triggered. An error or removed document stops the progressbar instantly.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public int</code> Progressbar</td>
<td>Value of progress bar</td>
</tr>
</tbody>
</table>

Example:

```
delegate void Progresshelper(object sender, ProgressbarDataEvent e);
private void yourProgressbar(object sender, ProgressbarDataEvent e)
{
    if (InvokeRequired)
    {
        Progresshelper pro = new Progresshelper(yourProgressbar);
        this.Invoke(pro, sender, e);
    }
    else
    {
        progressBar1.Value = e.Progressbar;
    }
}
```
4.3.5 Connection to the Device

Create an instance of the class `StartDevice`.

```java
StartDevice start = new StartDevice();
```

With the new object `start`, the function `initialization` is available.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `Error.ERROR_STATE` initialization(`OnDataEvent` dataEvent) | Initializes the COM ports according the settings in the file “Settings.xml”
return value:
`Error.ERROR_STATE.SUCCESS` = successful
`Error.ERROR_STATE.COM_PORT_SCANNER` = scanner port error
`Error.ERROR_STATE.VCOMS_SERVICE` = PKTVCOMS service not running
`Error.ERROR_STATE.VCOMP_SERVICE` = PKTVCOMP service not running
`Error.ERROR_STATE.COM_PORT_PKT4000` = PKT port error
`Error.ERROR_STATE.INVALID_VALUE` = dataEvent error
`Error.ERROR_STATE.UNKNOWN` = other errors |
| `Error.ERROR_STATE` initialization(`string` ScanPort, `string` PKTPort, `OnDataEvent` dataEvent) | Initializes the COM ports according the parameters
return value:
`Error.ERROR_STATE.SUCCESS` = successful
`Error.ERROR_STATE.COM_PORT_SCANNER` = scanner port error
`Error.ERROR_STATE.VCOMS_SERVICE` = PKTVCOMS service not running
`Error.ERROR_STATE.VCOMP_SERVICE` = PKTVCOMP service not running
`Error.ERROR_STATE.COM_PORT_PKT4000` = PKT port error
`Error.ERROR_STATE.INVALID_VALUE` = dataEvent error
`Error.ERROR_STATE.UNKNOWN` = other errors |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error.ERROR_STATE initialization(string strPort, OnDataEvent dataEvent)</td>
<td>Initializes the COM port only for working with VCOMP service&lt;br&gt;Return value:&lt;br&gt;Error.ERROR_STATE.SUCCESS = successful&lt;br&gt;Error.ERROR_STATE.COM_PORT_SCANNER = scanner port error&lt;br&gt;Error.ERROR_STATE.VCOMP_SERVICE = PKTVCOMP service not running&lt;br&gt;Error.ERROR_STATE.COM_PORT_PKT4000 = PKT port error&lt;br&gt;Error.ERROR_STATE.INVALID_VALUE = dataEvent error&lt;br&gt;Error.ERROR_STATE.UNKNOWN = other errors</td>
</tr>
<tr>
<td>void ClosePKT4000DLL()</td>
<td>Close DLL and free all resources.&lt;br&gt;Return value:&lt;br&gt;Error.ERROR_STATE.SUCCESS = successful&lt;br&gt;Error.ERROR_STATE.INVISIBLE_VALUE = background worker not running&lt;br&gt;Error.ERROR_STATE.UNKNOWN = other errors</td>
</tr>
</tbody>
</table>
4.3.5.1 initialization

Error.ERROR_STATE initialization(OnDataEvent dataEvent)

The method with one parameter has default values for the COM Ports according the file “Settings.xml”. Users can change the settings in this file. The file will be found e. g. for Windows 7 in the directory “C:\Users\username\AppData\Roaming\PrehKeyTec\PKT4000”.

Note: The COM Port value (string ScanPort) for MCI 111 A / ML 2 A is not used.

Example:

Error.ERROR_STATE ret = start.initialization(CustomEvent);
if (ret != Error.ERROR_STATE.SUCCESS)
    throw new PKT4000Exception(ret, "ERROR open device");

Error.ERROR_STATE initialization(string ScanPort, string PKTPort, OnDataEvent dataEvent)

If the method with three parameters is called, the user set the names of the COM ports in the parameter.

Example:

Error.ERROR_STATE ret = start.initialization("COM5", "COM29", CustomEvent);
if (ret != Error.ERROR_STATE.SUCCESS)
    throw new PKT4000Exception(ret, "ERROR open device");

Error.ERROR_STATE initialization(string strPort, OnDataEvent dataEvent)

If the method with two parameters is called, the user set the names of the VCOMP port in the parameter. VCOMP is a PrehKeyTec service and runs in the background.

Example:

Error.ERROR_STATE ret = start.initialization("COM28", CustomEvent);
if (ret != Error.ERROR_STATE.SUCCESS)
    throw new PKT4000Exception(ret, "ERROR open device");
4.3.5.2 ClosePKT4000DLL

void ClosePKT4000DLL()
Close the PKT4000DLL and free resources.

Example:
start.ClosePKT4000DLL();
### 4.4 PKT4000 Methods

The following methods are available:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error.ERROR_STATE CaptureImage(OnDataEvent event)</td>
<td>Capture a picture of the document. Only for PKT 4000.</td>
</tr>
<tr>
<td>Error.ERROR_STATE Product_Version()</td>
<td>Read the version of the PKT 4000.</td>
</tr>
<tr>
<td>Error.ERROR_STATE setRFID(bool b_RFID)</td>
<td>Enable/Disable RFID reading. Only for PKT 4000.</td>
</tr>
<tr>
<td>bool getCaptureImage()</td>
<td>Read value CaptureImage from file “Settings.xml”</td>
</tr>
<tr>
<td>Error.ERROR_STATE setCaptureImage(bool b_Capture)</td>
<td>If an OCRDataEvent was triggered, the CaptureImage method takes a picture from the current document on the scanner. Only for PKT 4000.</td>
</tr>
<tr>
<td>bool getFaceDetection()</td>
<td>Read value FaceDetection from file “Settings.xml”</td>
</tr>
<tr>
<td>Error.ERROR_STATE setFaceDetection(bool b_Detect)</td>
<td>Searches for a face of the captured image created by the CaptureImage method. This function is available, if the application directory contains the opencv DLL files. Only for PKT 4000.</td>
</tr>
<tr>
<td>bool getCreateImageFiles()</td>
<td>Read value CreateFiles from file “Settings.xml”</td>
</tr>
<tr>
<td>Error.ERROR_STATE setCreateImageFiles(bool b_createFiles)</td>
<td>If the value is set to true, captured images are automatically saved in the defined directory (read picture_folder from Settings.xml) under the defined name (default = &quot;ImageScan.jpg&quot; or set with function setFilename(string Filename)). Only for PKT 4000.</td>
</tr>
<tr>
<td>Error.ERROR_STATE setImageRotation(bool b_ImageRotation)</td>
<td>Set image rotation in “Settings.xml”. Rotates the image by 180 degrees after next scan. Only for PKT 4000.</td>
</tr>
<tr>
<td>bool getImageRotation()</td>
<td>Read value ImageRotate from file</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>setFilename(string Filename)</code></td>
<td>All OCR data fields can be used for creating the filename. It only works for documents with OCR. The following parameter can create a filename: passportnr, surname, givenname, type, code, nationality, dateofbirth, dateofexpiry and sex. The parameters are separated with a comma. The method <em>createImageFiles</em> must be set to true. Only for PKT 4000.</td>
</tr>
<tr>
<td><code>Load_Scanner_Default()</code></td>
<td>Set scanner to default values. Only for PKT 4000.</td>
</tr>
<tr>
<td><code>getPicturePath()</code></td>
<td>Shows the current directory where the images are saved.</td>
</tr>
<tr>
<td><code>setPicturePath(string path)</code></td>
<td>Sets the path of the saved images. Only for PKT 4000.</td>
</tr>
<tr>
<td><code>getXMLSettingPath()</code></td>
<td>Shows the current directory where the xml file is saved.</td>
</tr>
<tr>
<td><code>setProgressbar(bool Progress)</code></td>
<td>Starts a progressbar if a RFID event is triggered. Only for PKT 4000.</td>
</tr>
<tr>
<td><code>EnableScanning(bool enable, string delay)</code></td>
<td>Activate and deactivate scanning process. Only for PKT 4000.</td>
</tr>
<tr>
<td><code>ReadRFIDwithMRZ(string documentNumber, string dateOfBirth, string dateOfExpiry)</code></td>
<td>Read DG1 and DG2 from RFID chip on the basis of document number, date of birth and date of expiry. Only for PKT 4000.</td>
</tr>
<tr>
<td><code>ReadRFIDwithCAN(string CAN)</code></td>
<td>Read DG1 and DG2 from RFID chip on the basis of the Card Access Number (CAN). Only for PKT 4000. This function is not supported.</td>
</tr>
</tbody>
</table>
### 4.4.1 CaptureImage (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>Error.ERROR_STATE CaptureImage(OnDataEvent ScanEvent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>OnDataEvent ScanEvent</td>
</tr>
</tbody>
</table>
| return value | Error.ERROR_STATE.INVALID_VALUE  
Error.ERROR_STATE.COM_PORT_SCANNER  
Error.ERROR_STATE.UNSUPPORTED  
Error.ERROR_STATE.SUCCESS |
| Sample | pktMethods.CaptureImage(events); |
| Description | Takes a picture of the document and send it via event function. |

### 4.4.2 setPicturePath (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>Error.ERROR_STATE setPicturePath(string path)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>string path, name of path.</td>
</tr>
</tbody>
</table>
| return value | Error.ERROR_STATE.INVALID_VALUE  
Error.ERROR_STATE.UNSUPPORTED  
Error.ERROR_STATE.SUCCESS |
| Sample | string _picturepath = Environment.CurrentDirectory;  
pktMethods.setPicturePath(_picturepath); |
| Description | Set the path of saved images. |

### 4.4.3 getPicturePath

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>string getPicturePath()</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>none</td>
</tr>
<tr>
<td>return value</td>
<td>string path name</td>
</tr>
<tr>
<td>Sample</td>
<td>string path = pktMethods.getPicturePath();</td>
</tr>
<tr>
<td>Description</td>
<td>Get path of directory for saved images.</td>
</tr>
</tbody>
</table>

### 4.4.4 getXMLSettingPath

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>string getXMLSettingPath()</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>none</td>
</tr>
<tr>
<td>return value</td>
<td>string path name</td>
</tr>
<tr>
<td>Sample</td>
<td>string xpath = pktMethods.getXMLSettingPath();</td>
</tr>
<tr>
<td>Description</td>
<td>Get path of directory for XML files.</td>
</tr>
</tbody>
</table>
### 4.4.2 setCaptureImage (only for PKT 4000)

**Syntax:**
```
Error.ERROR_STATE setCaptureImage(bool b_Capture)
```

**Parameter**
- `true` = Enable capturing image, `false` = Disable capturing image

**Return Value**
- `Error.ERROR_STATE.INVALID_VALUE`
- `Error.ERROR_STATE.UNKNOWN`
- `Error.ERROR_STATE.UNSUPPORTED`
- `Error.ERROR_STATE.SUCCESS`

**Sample**
```
pktMethods.setCaptureImage(true);
```

**Description**
Enable or disable automatically capturing an image of the document on the scan window after reading OCR data. This function only works for documents with OCR data.

### 4.4.3 getCaptureImage

**Syntax:**
```
bool getCaptureImage()
```

**Parameter**
None

**Return Value**
- `true` or `false`

**Sample**
```
pktMethods.getCaptureImage();
```

**Description**
Read value “CaptureImage” from “Settings.xml”.

### 4.4.4 setFaceDetection (only for PKT 4000)

**Syntax:**
```
Error.ERROR_STATE setFaceDetection(bool b_Detect)
```

**Parameter**
- `true` = Enable Face Detection, `false` = Disable Face Detection

**Return Value**
- `Error.ERROR_STATE.INVALID_VALUE`
- `Error.ERROR_STATE.UNKNOWN`
- `Error.ERROR_STATE.UNSUPPORTED`
- `Error.ERROR_STATE.SUCCESS`

**Sample**
```
pktMethods.setFaceDetection(true);
```

**Description**
Enable or disable Face Detection of the document on the scan window. The image is sent when the trigger of ScanData appears. You need special DLL files in the application directory (e.g. “opencv.dll” etc.). The needed files are in the installation sub directory “lib”.

### 4.4.5 getFaceDetection

**Syntax:**
```
bool getFaceDetection()
```

**Parameter**
None

**Return Value**
- `true` or `false`

**Sample**
```
pktMethods.getFaceDetection();
```

**Description**
Read value “FaceDetection” from “Settings.xml”.
### 4.4.6 setCreateImageFiles (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>Error.ERROR_STATE setCreateImageFiles (bool b_createFiles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>true=Save an image, false=Don’t save an image</td>
</tr>
<tr>
<td>return value</td>
<td>Error.ERROR_STATE.INVALID_VALUE</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNKNOWN</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.SUCCESS</td>
</tr>
</tbody>
</table>

**Sample**

```
pktMethods.setCreateImageFiles (true);
```

**Description**
The captured image is saved in the directory, read from “Settings.xml” or the application directory (no path in XML file) under the name of “ImageScan.jpg” (default, will be overwritten by next scan) or name set by function setFilename(string Filename).

### 4.4.7 getCreateImageFiles

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>bool getCreateImageFiles()</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>None</td>
</tr>
<tr>
<td>return value</td>
<td>true or false</td>
</tr>
</tbody>
</table>

**Sample**

```
pktMethods.getCreateImageFiles();
```

**Description**
Read value “CreateFiles” from “Settings.xml”.

### 4.4.8 setFilename (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>Error.ERROR_STATE setFilename(string Filename)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>string Filename, name of created file, Take the key words passportnr, surname, givenname, type, code, nationality, dateofbirth, dateofexpiry and sex. You can also define your own Filename.</td>
</tr>
<tr>
<td>return value</td>
<td>Error.ERROR_STATE.INVALID_VALUE</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNKNOWN</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.SUCCESS</td>
</tr>
</tbody>
</table>

**Sample**

```
pktMethods.setFilename("passportnr,surname,givenname");
pktMethods.setFilename("W7GCH9ZY2_ANDERSON_JANE");
```

**Description**
All OCR data fields can be used for creating the filename. Only works for documents with OCR. The following parameters can create a filename: passportnr, surname, givenname, type, code, nationality, dateofbirth, dateofexpiry and sex. The parameters are separated with a comma. Filename can also be user defined. The method createImageFiles must be set to true. If face detection is turned on, an additional file with F_Filename (e.g. "F_W7GCH9ZY2_ANDERSON_JANE") is created.
### 4.4.9 setImageRotation (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>Error.ERROR_STATE setImageRotation(bool b_Rotate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>true=Rotate image by 180 degree, false=Rotate image by 360 degree</td>
</tr>
<tr>
<td>return value</td>
<td>Error.ERROR_STATE.INVALID_VALUE</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNKnown</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.SUCCESS</td>
</tr>
</tbody>
</table>

Sample: pktMethods.setImageRotation(true);

Description: Set image rotation in “Settings.xml”. Rotates the image by 180 degrees after next scan.

### 4.4.10 getImageRotation

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>bool getImageRotation()</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>None</td>
</tr>
<tr>
<td>return value</td>
<td>true or false</td>
</tr>
</tbody>
</table>

Sample: pktMethods.getImageRotation();

Description: Read value “ImageRotate” from “Hardware_Settings.xml”.

### 4.4.11 setProgressbar (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>void setProgressbar(bool b_Progress)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>bool b_Progress, b_Progress==true -&gt; enable progress bar</td>
</tr>
<tr>
<td>return value</td>
<td>Error.ERROR_STATE.SUCCeSS</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNKNOWN</td>
</tr>
</tbody>
</table>

Sample: pktMethods.setProgressbar(true);

Description: Starts a progressbar if a RFID event is triggered.

### 4.4.12 setRFID (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>Error.ERROR_STATE setRFID(bool On)</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>true=Enable RFID, false=Disable RFID</td>
</tr>
<tr>
<td>return value</td>
<td>Error.ERROR_STATE.INVALID_VALUE</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNKnown</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.RFID</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.UNSUPPORTED</td>
</tr>
<tr>
<td></td>
<td>Error.ERROR_STATE.SUCCeSS</td>
</tr>
</tbody>
</table>

Sample: pktMethods.setRFID(true);

Description: Enable or disable RFID functionality. Only available for PKT 4000 with RFID reader.
### 4.4.13 Product_Version

Syntax:  

\[
\text{Error.\textit{ERROR\_STATE} Product\_Version()}
\]

Parameter:  

None

Return Value:  

\[
\begin{align*}
\text{Error.\textit{ERROR\_STATE.COM\_PORT\_PKT4000}} \\
\text{Error.\textit{ERROR\_STATE.SUCCESS}}
\end{align*}
\]

Sample:  

pktMethods.Product_Version();

Description:  

Reads the version of the PKT 4000 and trigger the version event.

### 4.4.14 Load_Scanner_Default (only for PKT 4000)

Syntax:  

\[
\text{Error.\textit{ERROR\_STATE} Load\_Scanner\_Default()}
\]

Parameter:  

None

Return Value:  

\[
\begin{align*}
\text{Error.\textit{ERROR\_STATE.COM\_PORT\_SCANNER}} \\
\text{Error.\textit{ERROR\_STATE.INVALID\_VALUE}} \\
\text{Error.\textit{ERROR\_STATE.UNSUPPORTED}} \\
\text{Error.\textit{ERROR\_STATE.SUCCESS}}
\end{align*}
\]

Sample:  

pktMethods.Load_Scanner_Default();

Description:  

Set scanner to default values.

### 4.4.15 EnableScanning (only for PKT 4000)

Syntax:  

\[
\text{Error.\textit{ERROR\_STATE} EnableScanning(\text{bool} enable, \text{string} delay)} \\
\text{Error.\textit{ERROR\_STATE} EnableScanning(\text{bool} enable)}
\]

Parameter:  

\[
\begin{align*}
\text{bool} \text{ enable}=\text{true}, \text{activate scanner, enable}=\text{false}, \text{deactivate scanner} \\
\text{string} \text{ delay}, \text{range “0” to “30000” (not longer supported)}
\end{align*}
\]

Return Value:  

\[
\begin{align*}
\text{Error.\textit{ERROR\_STATE.COM\_PORT\_SCANNER}} \\
\text{Error.\textit{ERROR\_STATE.COM\_PORT\_PKT4000}} \\
\text{Error.\textit{ERROR\_STATE.INVALID\_VALUE}} \\
\text{Error.\textit{ERROR\_STATE.UNSUPPORTED}} \\
\text{Error.\textit{ERROR\_STATE.SUCCESS}}
\end{align*}
\]

Sample:  

pktMethods.EnableScanning(true);

Description:  

Activate or deactivate scanner. User can control scanning process by activating or deactivating scanner. *The delay means the reread delay of reading the same document again.*
### 4.4.16 ReadRFIDwithMRZ (only for PKT 4000)

**Syntax:**

```csharp
Error.ERROR_STATE ReadRFIDwithMRZ(string documentNumber, string dateOfBirth, string dateOfExpiry)
```

**Parameter:**

- `documentNumber`: the document number
- `dateOfBirth`: the date of birth
- `dateOfExpiry`: the date of expiry

**Return Value:**

- `Error.ERROR_STATE.RFID`
- `Error.ERROR_STATE.INVALID_VALUE`
- `Error.ERROR_STATE.UNSUPPORTED`
- `Error.ERROR_STATE.SUCCESS`

**Sample:**

```csharp
pktMethods.ReadRFIDwithMRZ("1A2B3C4D5", "123456", "112233");
```

**Description:**

Reads the DG1 and DG2 from RFID chip on the basis of document number, date of birth and date of expiry.

### 4.4.17 ReadRFIDwithCAN (only for PKT 4000, not yet supported)

**Syntax:**

```csharp
Error.ERROR_STATE ReadRFIDwithMRZ(string CAN)
```

**Parameter:**

- `CAN`: the Card Access Number

**Return Value:**

- `Error.ERROR_STATE.RFID`
- `Error.ERROR_STATE.INVALID_VALUE`
- `Error.ERROR_STATE.UNSUPPORTED`
- `Error.ERROR_STATE.SUCCESS`

**Sample:**

```csharp
pktMethods.ReadRFIDwithMRZ("123456");
```

**Description:**

Reads the DG1 and DG2 from RFID chip on the basis of the CAN.

### 4.4.18 EnableOCR

**Syntax:**

```csharp
Error.ERROR_STATE EnableOCR(bool enable)
```

**Parameter:**

- `enable=true`, activate OCR, `enable=false`, deactivate OCR

**Return Value:**

- `Error.ERROR_STATE.COM_PORT_PKT4000`
- `Error.ERROR_STATE.INVALID_VALUE`
- `Error.ERROR_STATE.SUCCESS`

**Sample:**

```csharp
pktMethods.EnableOCR(true);
```

**Description:**

Activate or deactivate OCR reading for PKT 4000 or MCI 111 A.
### 4.4.19 `getEnableScanner` (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th><code>bool getEnableScanner()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>none</td>
</tr>
<tr>
<td>return value</td>
<td><code>true</code> or <code>false</code></td>
</tr>
<tr>
<td>Sample</td>
<td><code>pktMethods.getEnableScanner();</code></td>
</tr>
<tr>
<td>Description</td>
<td>Get enable/disable Scanner stored in the XML file.</td>
</tr>
</tbody>
</table>

### 4.4.20 `setEnableScanner` (only for PKT 4000)

<table>
<thead>
<tr>
<th>Syntax:</th>
<th><code>Error.ERROR_STATE setEnableScanner(bool enable)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td><code>bool enable=true</code>, activate Scanner, <code>false</code>, deactivate Scanner</td>
</tr>
<tr>
<td>return value</td>
<td><code>Error.ERROR_STATE.UNSUPPORTED</code>, <code>INVALID_VALUE</code>, <code>SUCCESS</code></td>
</tr>
<tr>
<td>Sample</td>
<td><code>pktMethods.setEnableScanner(true);</code></td>
</tr>
<tr>
<td>Description</td>
<td>Save flag for activate/deactivate Scanner in the XML file.</td>
</tr>
</tbody>
</table>

### 4.4.21 `getEnableOCR`

<table>
<thead>
<tr>
<th>Syntax:</th>
<th><code>bool getEnableOCR()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>none</td>
</tr>
<tr>
<td>return value</td>
<td><code>true</code> or <code>false</code></td>
</tr>
<tr>
<td>Sample</td>
<td><code>pktMethods.getEnableOCR();</code></td>
</tr>
<tr>
<td>Description</td>
<td>Get enable/disable OCR stored in the XML file.</td>
</tr>
</tbody>
</table>

### 4.4.22 `setEnableOCR`

<table>
<thead>
<tr>
<th>Syntax:</th>
<th><code>Error.ERROR_STATE setEnableOCR(bool enable)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td><code>bool enable=true</code>, activate OCR, <code>false</code>, deactivate OCR</td>
</tr>
<tr>
<td>return value</td>
<td><code>Error.ERROR_STATE.UNKNOWN</code>, <code>INVALID_VALUE</code>, <code>SUCCESS</code></td>
</tr>
<tr>
<td>Sample</td>
<td><code>pktMethods.setEnableOCR(true);</code></td>
</tr>
<tr>
<td>Description</td>
<td>Save flag for activate/deactivate OCR in the XML file.</td>
</tr>
</tbody>
</table>
### 4.4.23 getEnableMSR

**Syntax:**

```c
bool getEnableMSR()
```

**Parameter:**

None

**Return Value:**

true or false

**Sample:**

```c
pktMethods.getEnableMSR();
```

**Description:**
Get enable/disable MSR stored in the XML file.

### 4.4.24 setEnableMSR

**Syntax:**

```c
Error.ERROR_STATE setEnableMSR(bool enable)
```

**Parameter:**

bool enable=true, activate MSR, enable=false, deactivate MSR

**Return Value:**

Error.ERROR_STATE.UNKNOWN

Error.ERROR_STATE.INVALID_VALUE

Error.ERROR_STATE.SUCCESS

**Sample:**

```c
pktMethods.setEnableMSR(true);
```

**Description:**
Save flag for activate/deactivate MSR in the XML file.

### 4.4.25 getEnableRFID (only for PKT 4000)

**Syntax:**

```c
bool getEnableRFID()
```

**Parameter:**

None

**Return Value:**

true or false

**Sample:**

```c
pktMethods.getEnableRFID();
```

**Description:**
Get enable/disable RFID stored in the XML file.

### 4.4.26 setEnableRFID (only for PKT 4000)

**Syntax:**

```c
Error.ERROR_STATE setEnableRFID(bool enable)
```

**Parameter:**

bool enable=true, activate RFID, enable=false, deactivate RFID

**Return Value:**

Error.ERROR_STATE.UNKNOWN

Error.ERROR_STATE.INVALID_VALUE

Error.ERROR_STATE.SUCCESS

**Sample:**

```c
pktMethods.setEnableRFID(true);
```

**Description:**
Save flag for activate/deactivate RFID in the XML file.
4.4 DeviceLED (only for PKT 4000)

Create a new instance of the `DeviceLED.setLED` structure. Now the MSR, RFID, Scanner and Power status lights are available. The possible colours are red, green and orange. The lights can be turned on (value is true) or off (value is false).

Syntax:
```csharp
public struct setLED
{
    public bool MSRGreen;
    public bool MSROff;
    public bool MSROrange;
    public bool MSRRed;
    public bool PowerGreen;
    public bool PowerOff;
    public bool PowerOrange;
    public bool PowerRed;
    public bool RFIDGreen;
    public bool RFIDOff;
    public bool RFIDOrange;
    public bool RFIDRed;
    public bool ScannerGreen;
    public bool ScannerOff;
    public bool ScannerOrange;
    public bool ScannerRed;
}
```

Send the structure object via the `LEDCommand()` method to apply the status light settings.

Syntax: `bool LEDCommand(DeviceLED.setLED led);`

Example:
```csharp
private void LED_Click(object sender, EventArgs e)
{
    DeviceLED.setLED SetLED = new DeviceLED.setLED();
    SetLED.MSRGreen = true;
    SetLED.PowerOrange = true;
    SetLED.RFIDRed = true;
    SetLED.ScannerOff = true;
    // start is set earlier (e.g. constructor) as StartDevice start = new StartDevice();
    if( !start.LED.LEDCommand(SetLED) )
        throw new PKT4000Exception(Error.ERROR_STATE.COM_PORT_PKT4000, "ERROR set LED");
}
```
4.5 DeviceBuzzer

Create a new instance of the DeviceBuzzer.setBuzzer structure. Now the parameters of the structures are available:

- **UInt16** frequency (frequency of the tone)
- **byte** volume (volume from 1 to 100 %)
- **UInt16** duration (duration in 50 ms steps from 50 ms to 11150 ms)

Syntax:
```csharp
public struct setBuzzer
{
    public UInt16 frequency;
    public byte volume;
    public UInt16 duration;
}
```

Send the structure object via the BuzzerCommand() method to apply the buzzer settings.

Syntax: `Error.ERROR_STATE BuzzerCommand(DeviceBuzzer.setBuzzer buzzer);`

Example:
```csharp
private void Buzzer_Click(object sender, EventArgs e)
{
    DeviceLED.setBuzzer SetBuzzer = new DeviceLED.setBuzzer();
    SetBuzzer.duration = 1000;       // 1000 ms
    SetBuzzer.volume = 70;          // 70 %
    SetBuzzer.frequency = 1500;     // 1500 Hz
    // start is set earlier (e. g. constructor) as StartDevice start = new
    // StartDevice();
    Error.ERROR_STATE ret = start.Buzzer.BuzzerCommand(SetBuzzer);
    if( ret != Error.ERROR_STATE.SUCCESS )
        throw new PKT4000Exception(ret, "ERROR set LED");
    
}
4.6 PKT 4000 Exception

You can throw PKT 4000 specified exceptions.

```java
throw new PKT4000Exception(Error.ERROR_STATE.COM_PORT_PKT4000, "ERROR open device");
```

The parameters are the error code and the message string.

Catch the PKT4000 in the catch block.

```java
Example:

private void PKT4000Form_Load(object sender, EventArgs e)
{
    try
    {
        // connection to device
        Error.ERROR_STATE ret = start.initialization(CustomEvent);
        if (ret != Error.ERROR_STATE.SUCCESS)  // Default is "COM12" for Scanner
            throw new PKT4000Exception(ret, "ERROR open device");
    }
    catch (PKT4000Exception ex)
    {
        Application.DoEvents();
    }
    catch (Exception ex)
    {
        toolStripStatusLabel1.Text = ex.Message;
        Application.DoEvents();
    }
}
```