
IDParser Specification

IDParserQLibNet version 4.3.5



Copyright

Printed in the United States of America

3511 Silverside Rd.
Suite 105
Wilmington, DE 19810
www.TokenWorks.com

Copyright 2022 TokenWorks, Inc.

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of TokenWorks, Inc.

TokenWorks, IDWedge, IDVisor, CardTool, CardDB and DigiSwipe are trademarks of TokenWorks, Inc.

About This Functional Requirements Document

This document describes the software libraries (DLL's) used to parse field information encoded on ID cards and Drivers Licenses in the Windows environment.

Development Environment

The consuming application shall be developed in C# - VS2017 .Net 4.5.2 or later

Operating Environment

Target Platform

Win7-10, x86 – x64

Table of Contents

IDPARSERQLIBNET VERSION 4.3.5	1
COPYRIGHT	2
ABOUT THIS FUNCTIONAL REQUIREMENTS DOCUMENT.....	2
DEVELOPMENT ENVIRONMENT.....	2
OPERATING ENVIRONMENT	2
<i>Target Platform</i>	2
TABLE OF CONTENTS.....	3
SCANNER DLL PARSING SPECIFICATION.....	4
MAGLL DEFINITIONS	5
<i>Loyalty cards</i>	6
<i>BANK CARD (Credit Card) Fields</i>	6
MAGLL INTERFACE	7
PDF417 DEFINITIONS.....	8
PDF417 INTERFACE.....	10
TRIAL EVALUATION PERIOD.....	11
KEYRESPONSE.....	11
VALIDATEONSERVER API.....	12
REVISION INFORMATION.....	13

Scanner DLL parsing specification

This document describes the software libraries (DLL's) used to parse field information encoded on ID cards and Drivers Licenses in the Windows environment.

ID/DL cards with a magnetic stripe use the MagQLibNect.dll library and ID/DL cards with a 2D barcode use the PdfQLibNet.dll library. The data stream coming from the card reader is received by the application and sent to either the Magnetic or the Barcode library for parsing. The libraries decode the fields embedded on the card and return them as null terminated ASCII strings.

The parsing functions are protected by a license file stored in the same directory as the libraries. The name of the license file is: ***LicenseFile.xml***

An application example is provided with the DLL's which is written in C# and developed within the Microsoft Visual studio VS2017 .NET environment. The sample project is called ***IDarser***

MAGLL definitions

Namespace definition:

using MagQLibNet; //magstripe library

Class:

Magstripe

Public fields

Magstripe.titleP_DL
Magstripe.firstnameP_DL
Magstripe.midnameP_DL
Magstripe.lastnameP_DL
Magstripe.addrP_DL
Magstripe.cityP_DL
Magstripe.stateP_DL
Magstripe.zipP_DL
Magstripe.dIP_DL
Magstripe.expP_DL
Magstripe.dobyearP_DL
Magstripe.dobMMDDP_DL
Magstripe.ClassP_DL
Magstripe.RestrictionsP_DL
Magstripe.EndorsementsP_DL
Magstripe.SexP_DL
Magstripe.HeightP_DL
Magstripe.WeightP_DL
Magstripe.HairP_DL
Magstripe.EyeP_DL
Magstripe.ScannerSN // serial number of scanner
Magstripe.track1 // ASCII string for track 1
Magstripe.track2 // ASCII string for track 2
Magstripe.track3 // ASCII string for track 3
Magstripe.Version // DLL revision string
Magstripe.firstname_CC //Credit Card Fields
Magstripe.lastname_CC
Magstripe.PAN_CC
Magstripe.EXPMM_CC
Magstripe.EXPY_Y_CC
Magstripe.IsoBin_DL
Magstripe.IssuingState_DL
Magstripe.KeyResponse //Integer return code of License
Magstripe.KeyStatus //String return status of License

Method:

`int ParseDL(byte[]MagBuff)`

The application will send the entire buffer of data returned from swiping a magnetic stripe card to the ParseDL method. The buffer sent must be dynamically sized for each swipe to ensure buffers do not overflow. Based on the return value, the public fields will be available. The application should define string variables for each public field, and an integer for the return value of the ParseDL method.

Return Values:

`Int Problem = ParseDL(byte[]MagBuff);`

No Problem parsing:	Problem = 0
Track1 fails to parse:	Problem = 1
Track2 fails to parse:	Problem = 2
Track3 fails to parse:	Problem = 4
Unparsed Loyalty card	Problem = 86
buffer fails to divide into 3 tracks	Problem = -1
Exception	Problem = 4096

If the return value indicates Track2 fails to parse or Non ID/DL card, then the ASCII string data (`Magstripe.track#`) for each track that is available can be used by the application. If the card reader/scanner is not correctly configured with the licensing key then the return value of 99 is returned and no data is available.

The return value is a logical OR of all the return codes

Loyalty cards

The following cards will return an 86 and not be parsed:

Track2 Starts With	Identified as
;4290	AAA loyalty card
;4381	AAA loyalty card
;4382	AAA loyalty card
;5490	AAA loyalty card
;6202	AAA loyalty card

BANK CARD (Credit Card) Fields

The most common bank cards are identified using the following criteria:

Track2 Starts With	Identified as
;4	Visa
;34	Amex
;37	Amex
;51	Master Card
;52	Master Card
;53	Master Card
;54	Master Card
;55	Master Card
;6011	Discover

MAGLL Interface

The MagQLibNet library supports a COM visible interface

```
[ComVisible(true)]
public interface IParsedMAG
{
    int problem_ID { get; }
    int keyresponse_ID { get; }
    string keyrespstr { get; }
    string Title_ID { get; }
    string FirstName_ID { get; }
    string Mid_ID { get; }
    string Last_ID { get; }
    string Addr_ID { get; }
    string Addr2_ID { get; } //returns empty string, missing from Mag DLs
    string City_ID { get; }
    string State_ID { get; }
    string Zip_ID { get; }

    string DL_ID { get; }
    string Exp_ID { get; }
    string DobYear_ID { get; }
    string DobMMDD_ID { get; }
    string DocIssDate_ID { get; } //returns empty string, missing from Mag DLs

    string Height_ID { get; }
    string Wt_ID { get; }
    string Eye_ID { get; }
    string Sex_ID { get; }
    string Hair_ID { get; }

    string DLClass_ID { get; }
    string Rest_ID { get; }
    string Endr_ID { get; }
    string Country_ID { get; } //returns empty string, missing from Mag DLs
    string Race_ID { get; } //returns empty string, missing from Mag DLs

    string SSN_ID { get; } //returns empty string, missing from Mag DLs
    string Rank_ID { get; } //returns empty string, missing from Mag DLs
    string ScanSN_ID { get; }
    string RealID_ID { get; } //returns empty string, missing from Mag DLs
    string IsoBin_ID { get; }
    string IssuingState_ID { get; }
}
```

PDF417 definitions

Namespace definition:

```
using PDFDLL; //pdf417 barcode library
```

Class:

Barcode

Public Fields

```
Barcode.State_DL  
Barcode.City_DL  
Barcode.Addr_DL  
Barcode.First_DL  
Barcode.Last_DL  
Barcode.Mid_DL  
Barcode.Title_DL  
Barcode.DI_DL  
Barcode.Exp_DL  
Barcode.DobYear_DL  
Barcode.DobMMDD_DL  
Barcode.Zip_DL  
Barcode.Height_DL  
Barcode.Eye_DL  
Barcode.Sex_DL  
Barcode.Class_DL;  
Barcode.Rest_DL;  
Barcode.Endr_DL;  
Barcode.Wt_DL;  
Barcode.Hair_DL;  
Barcode.Country_DL;  
Barcode.Race_DL;  
Barcode.DocIssDate_DL;  
Barcode.Addr2_DL;  
Barcode.Rank_DL; // Military ID  
Barcode.RealID_DL; // F=Fully Compliant, N= Not Compliant  
Barcode.SSN_DL; // Optional pre 2012 Military ID  
Barcode.PDF_String // Ascii string for 2D barcode  
Barcode.KeyResponse //Integer return code of License  
Barcode.KeyStatus //String return status of License  
Barcode.KeyswMaintRenewDate  
Barcode.IsoBin_DL  
Barcode.IssuingState_DL
```

Method:

```
int Parse417(byte[]buff)
```

The application will send the entire buffer of data returned from scanning a barcard to the Parse417 method. The buffer sent must be dynamically sized for each swipe to ensure buffers do not overflow. Based on the return value, the public fields will be available. The application should define string variables for each public field, and an integer for the return value of the Parse417 method.

Return Values:

Int Problem = Parse417(byte[]buff);

No Problem parsing:	Problem = 0
No ID/DL#:	Problem = 1
NAME issue:	Problem = 2
No Address:	Problem = 4
No City:	Problem = 8
No State:	Problem = 16
No Zip:	Problem = 32
No Height:	Problem = 64
No Eye:	Problem = 128
No Sex:	Problem = 256
No EXP	Problem = 512
NO DOB:	Problem = 1024
NO DATA	Problem = 2048
Exception	Problem = 4096

If the return value indicates NO DATA, then the ASCII string data ([Barcode.PDF_String](#)) is available and can be used by the application. If the card reader/scanner is not correctly configured with the licensing key then the return value of 99 is returned and no data is available.

When calling the Parse417 method, you must use a try/catch statement to protect the DLL from exceptions caused by unknown or incorrectly encoded ID/DL cards.

The return value is a logical OR of all the return codes

PDF417 Interface

[ComVisible(true)]

```
public interface Iparsed417
{
    int problem_ID { get; }
    int keyresponse_ID { get; }
    string keyrespstr_ID { get; }
    string Title_ID { get; }
    string FirstName_ID { get; }
    string Mid_ID { get; }
    string Last_ID { get; }
    string Addr_ID { get; }
    string Addr2_ID { get; }
    string City_ID { get; }
    string State_ID { get; }
    string Zip_ID { get; }
    string DI_ID { get; }
    string Exp_ID { get; }
    string DobYear_ID { get; }
    string DobMMDD_ID { get; }
    string DocIssDate_ID { get; }
    string Height_ID { get; }
    string Wt_ID { get; }
    string Eye_ID { get; }
    string Sex_ID { get; }
    string Hair_ID { get; }
    string DLClass_ID { get; }
    string Rest_ID { get; }
    string Endr_ID { get; }
    string Country_ID { get; }
    string Race_ID { get; }
    string SSN_ID { get; }
    string Rank_ID { get; }
    string ScanSN_ID { get; }
    string ReallID_ID { get; }
    string IsoBin_ID { get; }
    string IssuingState_ID { get; }
}
```

Trial Evaluation period

The Trial licensed version of the DLLs has a 10 day evaluation period. The parsed fields available during the trial will be: First, Mid, Last, Address, City, and State. The remainder parsed fields will have the 1st and every 3rd character displayed as an *. Example, if the zip code is 12345, the output will be *23*5

KeyResponse

The KeyResponse has been added to help in determining the state of the license; the returned field is available in both MagQLibNet and PdfQLibNet and can be used to evaluate the performance of the licensing system. There is also a KeyStatus string that provides additional information about the license in string form, retrieved from the license server.

Description	KeyResponse	
License is valid	0	
License is invalid	99	
Version Mismatch / sw maint exp	98	
Refresh Required	97	(Currently un-used, RFU)
License is in Trial mode	96	

Note: if the KeyResponse is not 0 or 96, then every field will have the 1st and 3rd character scrambled.

The Key Response code will be updated on load of the DLL. The application should have error handlers for each of these codes and take specific action as required.

The Key Status is a string that can help decipher the key response, additional information will be provided with this string from the license server, for example: The date the license expires. This string should be logged in non- GUI deployments to assist in maintaining the license.

When there is a key detection error, the Key Response = 99 will display, normally this occurs when the license file is missing, or if the key has expired. You will have to re-activate the license when this is returned, either with the same or a new Activation Code, or, after making arrangements to allow the current key to be returned into service.

ValidateOnServer API

Version 4.3.5 takes the default state of the connection to the license server to Off-Line. To compensate for the disconnected state, an API was developed to allow for the consuming application to manually connect /disconnect from the license server. The primary reason to have a connection available to the license server is to allow for updating the license file with new subscription information. The s/w Maintenance date is used to provide a subscription like behavior, when you renew your s/w maintenance date with Tokenworks, you will need to connect to the license server to download a new license file, and this API supports the connecting and disconnecting to the license server. NOTE: For PdfLibNet.dll (Barcodes) this API is run automatically approximately 1 minute after boot, to check for a license update. You would only need to call this API if your project consisted of only the MagLibNet.dll.

The API call is defined:

```
bool ValidateOnServer(bool OnOff)
```

This license file must be loaded first prior to making this call:

```
namespace ParserDemo
{
    public partial class frmMain : Form
    {
        PdfSrvrSV.Barcode bc;
        bool showActivation; //used to control the use of the Activation form

        showActivation true; // pass this to the constructor, for gui based demo app
        bc = new PdfSrvrSV.Barcode(showActivation);
        bool pdfLib = bc.ValidateOnServer(true);
    }
}
```

Note: `bool MagLib = mc.ValidateOnServer(true);` // MagSrvrSV.dll also supports the API

Revision Information

EVOLUTION OF THE DOCUMENT				
Version	Version	Date	Author	Evolution
1.0	SP-032310-PC-R0	3/23/10	PC	First Release
1.5	SP-102311-PC-R1	10/23/12	PC	Add Country and Race to PDF417
1.9	SP-043013-PC-R2	04/30/13	PC	Add Document Issue Date to PDF417
2.0	SP-09062013-R3	09/06/13	PC	Add 2 nd Address line to PDF417
2.8	SP-12182014-R4	12/18/14	PC	Add credit card parsing to MAGDLL
2.9	SP-12112015 – R5	12/11/15	PC	Update PDF417 DOB_MN and DCT middle name parsing
3.0	SP-06082016 –R6	06/08/16	PC	Make every 3 rd char = * during 30 evaluation period
3.3	SP-05102018-R7	05/10/18	PC	Add Key Response field, update document to reflect new project names.
3.4	SP-11132018-R8	11/13/18	PC	Add Title parsing to DCT field, target .net 2.0
3.5	SP-12102018-R9	12/10/18	PC	Add Interface to libraries
3.6	SP-03272019-R10	03/27/19	PC	Add new MN Magstripe and add gender X to AAMVA barcode
3.7	SP-05072019-R11	05/07/19	PC	Fix BC city name>13, improve field parsing and full name parsing
3.8	SP-12162019-R12	12/16/19	PC	Update last name/title parser to work with last name with spaces
3.9	SP-07172020-R-13	07/17/20	PC	Add RealID to Barcode
4.3	SP-10282021-R14	10/28/21	PC	New NameSpaces MagQLibNet PdfQLibNet, improve formatting
4.3.5	SP12132022-R15	12/13/22	PC	Add IsoBin_DL and IssuingState_DL and add ValicateOnServer API