User's Guide

Table of Contents

1. Introduction.

Encoding SDK (DMPS_E) is a part of the **Data Matrix Protection Suite (DMPS)** - easy-to-use software package that allows not only to encode/decode text information into/from a Data Matrix symbol, but to digitally protect this symbol from counterfeiting or tampering with, as well. With **DM Protection Suite**, protecting your products and documents has never been quicker or easier!

DMPS consists of the 2 software packages – encoding (**DMPS_E)** and decoding (**DMPS_D)**. The Encoding software has a proprietary built-in mechanism allowing either to create digital signature (enable Authentication) to the encoded Data Matrix symbol or to encrypt it. The Decoding software decrypts the symbol and/or checks it for authenticity while extracting the information encoded into it.

Depending on the application, the packages can be used either together or separately. In a Supply Chain application, for example, the encoding software can be used on the "manufacturing end" and the decoding software – on "receiving end".

1.1 Scope

This document is applicable to the **Data Matrix Protection Suite - Encoding** SDK.

SDK is notated as **DMPS_E_v.x**.x

Library interface is built for Windows (XP …10)/64. Windows 32-bit and Linux 32/64 versions are available per customer request. Only dynamic library is available.

The library is designed to:

- encode data into Data Matrices ECC200 in accordance with ISO/IEC 16022 Symbology specification; and
- enable Authentication capabilities for Data Matrix symbol utilizing proprietary 2DTG's authentication/encryption mechanism (User ID/**Authentication key**); and/or
- encrypt Data Matrix symbol with customer defined **Encryption Key (Product ID).**

DMPS_E is based on the latest 2DTG's Data Matrix Encoding library and can be used both for the DM protection purposes, as it's described in this User's Guide, and regular encoding.

1.2 References

- ISO/IEC 16022 - Symbology specification - Data Matrix
- U.S. Patent No.: 8,297,510 B1 "Mathematical method of 2D barcode authentication and protection for embedded processing"
- 2DTG "Data Matrix Encoding SDK"

1.3 SDK composition

Trial/Evaluation DMPS_E SDK package – fully functional 30-days trial version downloadable from 2DTG site:

- Windows DLL (**ECC200.DLL**) to encode/authenticate/encrypt Data Matrix symbol.
- C# source code of the Demo program **EncodeDemo.exe** and C++ (MSVC 2017) source code of the Demo program **VSCppDemo.exe,** illustrating the DLL usage.
- Sample **Authentication Key** (**User.id** – file) for programs evaluation.
- "**ReadMe -** file"– description of the program evaluation (includes DEMO UserID (**Authentication key**), DEMO Product ID (**Encryption key**)).
- **DMPS_Samples** (generated as illustration of the program operation in different modes):
  - NotProt.bmp (not protected)
  - USig.bmp (protected with User ID - Data Matrix Authentication is enabled)
  - PSig.bmp (protected with Product ID - Data Matrix Encryption)
  - USigPSig.bmp (protected both with User ID and Product ID - combined Data Matrix authentication and encryption)

Operational package includes:

- Licensed DMPS_E SDK – includes 1 (one) free Development license if customer purchases DMPS_D licenses.
- Unique Authentication key/UserID – 2DTG can provide multiple IDs per customer request if customer purchases DMPS_D licenses.

OEM Operational package includes:

- Licensed DMPS_E SDK – includes 5 (five) free Development licenses
- Unique Authentication key/UserID – 2DTG can provide multiple IDs per customer request if customer purchases DMPS_D licenses
- Authentication key/UserID Generator (per OEM customer purchasing DMPS_D licenses)

2. Library Interface.

```
//Signer.h 2011.01.18
#ifndef SIGNER_H
#define SIGNER_H
                // --------------- //
                // --------------- //
                //  DLL interface  //
                // --------------- //
                // --------------- //

  #define FName_EncLib "EncLib.dll"
  #define IE_DLL

  typedef void* PMatrix;
  typedef bool TMatrix[144][144];
  typedef unsigned char byt;

  enum T_DIMENSIONS{              // matrix sizes
   ms_auto    = -1,   ms_10_10   =  0,   ms_12_12   =  1,   ms_14_14   =  2,
   ms_16_16   =  3,   ms_18_18   =  4,   ms_20_20   =  5,   ms_22_22   =  6,
   ms_24_24   =  7,   ms_26_26   =  8,   ms_32_32   =  9,   ms_36_36   = 10,
   ms_40_40   = 11,   ms_44_44   = 12,   ms_48_48   = 13,   ms_52_52   = 14,
   ms_64_64   = 15,   ms_72_72   = 16,   ms_80_80   = 17,   ms_88_88   = 18,
   ms_96_96   = 19,   ms_104_104 = 20,   ms_120_120 = 21,   ms_132_132 = 22,
   ms_144_144 = 23,   ms_18_8    = 24,   ms_32_8    = 25,   ms_26_12   = 26,
   ms_36_12   = 27,   ms_36_16   = 28,   ms_48_16   = 29
  };

  const int dimentions_ess200[] =
  {10,10,   12,12,   14,14,   16,16,   18,18,   20,20,   22,22,   24,24,
   26,26,   32,32,   36,36,   40,40,   44,44,   48,48,   52,52,   64,64,
   72,72,   80,80,   88,88,   96,96,   104,104, 120,120, 132,132, 144,144,
   8,18,    8,32,    12,26,   12,36,   16,36,   16,48
  };
  #define RowDimention(dim) dimentions_ess200[2*dim]
  #define ColDimention(dim) dimentions_ess200[2*dim+1]
```

```
  enum T_COMPACTION_TYPES{      // matrix types
   encAuto     =    0,  // default
   encC40      =  230,  encBase256  =  231,  encANSIX12  =  238,
   encText     =  239,  encEDIFACT  =  240,  encASCII    =  254
  };

  char*strTCOD[] ={"Auto" ,"C40" ,"Base256" ,"ANSIX12" ,"Text" ,"EDIFACT"
,"ASCII",""};

  byt  TypeCOD[] =
{encAuto,encC40,encBase256,encANSIX12,encText,encEDIFACT,encASCII};

  #define Type2Str_ecc200(t,s) {s="Error TYPE";\
    for(int
_=0;strlen(strTCOD[_])>0;_++)if(t==TypeCOD[_]){s=strTCOD[_];break;} }
//---------------------------------------------------------------------------
// Symbol Errors
enum DM_SYMBOL_VALIDATION{
  DM_AU_OK        =  0,
  DM_AU_NA        =  1,   // Not Applicable
  DM_AU_FAILURE   =  2,   // Signature failure (in decoder only!)
  DM_AU_ERR_PID   =  7    // Wrong Product ID structure/length
};
//---------------------------------------------------------------------------
enum T_RESULT{
  DME_RES_OK        = 0,
  DME_RES_LONG1     = 1,   // Too long text for this matrix size
  DME_RES_LONG2     = 2,   // Too long text for all sizes
  DME_RES_ANSIX12   = 3,   // Unresolved character in ANSIX12 encodation
  DME_RES_EDIFACT   = 4,   // Unresolved character  for Edifact encodation
  DME_RES_AU_FAIL   = 5,   // Authentication failure
  DME_RES_TYPE      = 6,   // Type error
  DME_RES_DIMENSION = 7,   // Dimension error
  DME_RES_DLL       = 8,   // DLL error
  DME_RES_PARAMS    = 9,   // bug of parameters
  DME_RES_UNREGISTR =10,   // Unregistered version SDK
  DME_RES_UserID    =11,   // User ID not found
  DME_RES_UNKNOWN   =12    // Unknown error
};
//---------------------------------------------------------------------------

#define stdcall __stdcall

//---------------------------------------------------------------------------
extern "C" T_RESULT IE_DLL stdcall ConnectEncoder(); // first to be called

//
//============================================================================
//                      INTERFACE FUNCTIONS
//
//============================================================================
// encoding (with no protection)
extern "C" T_RESULT IE_DLL stdcall MakeMatrix_ecc200(
        byt*data, int len,      // data array, data length
        T_COMPACTION_TYPES cmp, // compaction type (encAuto - most compact)
```

```
        T_DIMENSIONS dim,       // index in T_DIMENSIONS (ms_auto- minimum
available)
        PMatrix &Matrix         // resulting Matrix
);

// ------------------------------------------------------------------------
// signing by UserID
extern "C" T_RESULT IE_DLL stdcall UserSign_ecc200(
        PMatrix Matrix    // (In/Out) Matrix
);

// ------------------------------------------------------------------------
// signing by ProductID
extern "C" T_RESULT IE_DLL stdcall ProductSign_ecc200 (
        PMatrix Matrix,   // (In/Out) Matrix
        char *ProductID   // string with ProductID (NULL: no Product
Signature)
);                        // returns true, if matrix succesfully signed

// ------------------------------------------------------------------------
// write BMP into file or Clipboard
extern "C" T_RESULT IE_DLL stdcall SaveMatrix_ecc200(
        PMatrix Matrix,         // Matrix
        int     modul,          // module size (>0)
        int     marge,          // space from the image margin (>=0)
        char   *fName           // File Name (NULL => write into Clipboard)
);

// ------------------------------------------------------------------------
//              supplementary functions:
// ------------------------------------------------------------------------
// translate result into character string
extern "C" void IE_DLL stdcall Result_ecc200(
        T_RESULT r,     // (IN) r
        char*b,int l    // (OUT)b - buffer for string, (OUT)l - length of b
);
// ------------------------------------------------------------------------
// return the DLL build date
extern "C" char* IE_DLL stdcall Version_ecc200();
// ------------------------------------------------------------------------

#endif  // SIGNER_H
```

3. Interface description

1. `typedef void* PMatrix;`
   A Data Matrix handler.

2. `typedef bool TMatrix[144][144];`

   User shell define the object of type TMatrix in his application. This object will represent the temporary Matrix Image. The Matrix Image is **output** parameter in function **`MakeMatrix_ecc200`**

It is the **input** and **output** parameter in functions
UserSign_ecc200
ProductSign_ecc200

3. **T_DIMENSIONS** - set of available matrix dimensions.
4. **T_COMPACTION_TYPES** – set of available compaction schemes.
5. **DM_SYMBOL_VALIDATION** – results of symbol signature validation.
6. **T_RESULT** – basic error codes in encoding.

7. The mean of procedures is clear from source code of C++ application below.

```
// EPR.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <Windows.h>
#include <direct.h>
#include "Signer.h"

PMatrix Matrix;

int _tmain(int argc, _TCHAR* argv[])
{
  HINSTANCE DllEPR=NULL;
  int result = 0;

  typedef T_RESULT (stdcall *TMakeMatrix )
           (byt*,int,T_COMPACTION_TYPES,T_DIMENSIONS,PMatrix &);
  typedef T_RESULT (stdcall *TUserSign   )(PMatrix);
  typedef T_RESULT (stdcall *TProductSign)(PMatrix,char*);
  typedef T_RESULT (stdcall *TSaveMatrix )(PMatrix,int,int,char*);
  typedef void     (stdcall *TResult     )(T_RESULT,char*,int);
  typedef char*    (stdcall *TVersion    )();

  TMakeMatrix  MakeMatrix  = NULL;
  TUserSign    UserSign    = NULL;
  TProductSign ProductSign = NULL;
  TSaveMatrix  SaveMatrix  = NULL;
  TResult      Result      = NULL;
  TVersion     Version     = NULL;

  DllEPR = LoadLibrary(L"..\\EPR.DLL");
  if(DllEPR==NULL){
    printf("Error DLL!\n");
    result = -1;
    goto exit;
  }
  MakeMatrix = (TMakeMatrix) GetProcAddress(DllEPR,"MakeMatrix_ecc200");
  UserSign   = (TUserSign)   GetProcAddress(DllEPR,"UserSign_ecc200");
  ProductSign= (TProductSign)GetProcAddress(DllEPR,"ProductSign_ecc200");
  SaveMatrix = (TSaveMatrix) GetProcAddress(DllEPR,"SaveMatrix_ecc200");
  Result     = (TResult)     GetProcAddress(DllEPR,"Result_ecc200");
  Version    = (TVersion)    GetProcAddress(DllEPR,"Version_ecc200");
```

```
if(  MakeMatrix == NULL || UserSign == NULL || ProductSign == NULL
  || SaveMatrix == NULL || Result   == NULL || Version     == NULL)
{
  FreeLibrary(DllEPR);
  printf("Function name error!\n");
  result = -1;
  goto exit;
}

{
  unsigned char text[] = "Greetings to everybody";
  T_COMPACTION_TYPES type = encAuto;
  T_DIMENSIONS size = ms_auto;
  int L = sizeof(text),

  rQ, // = dimentions_ess200[2*size],
  cQ; // = dimentions_ess200[2*size+1];
  T_RESULT r;

  r = MakeMatrix( text,L,type,size,Matrix);
  if(r != DME_RES_OK){
    result = r;
    printf("MakeMatrix Result = %d\n",result);
    goto exit;
  }

  r = SaveMatrix(Matrix, 2, 24, "Greet.bmp"); // NULL => Clipboard
  if(r != DME_RES_OK){
    result = r;
    printf("SaveMatrix Result = %d\n",result);
    goto exit;
  }
  printf("Greet.bmp saved!\n");

  r = UserSign(Matrix);
  if(r !=DME_RES_OK){
    result = r;
    printf("UserSign Result = %d\n",result);
     goto exit;
  }
  r = SaveMatrix(Matrix, 2, 24, "Greet_u.bmp"); //NULL);
  if(r != DME_RES_OK){
    result = r;
    goto exit;
  }
  printf("Greet_u.bmp saved!\n");

  r = ProductSign(Matrix, "s1");
  if(r != DME_RES_OK){
    result = r;
    printf("ProductSign Result = %d\n",result);
    goto exit;
  }
  r = SaveMatrix(Matrix, 2, 24, "Greet_u_s1.bmp"); //NULL);
  if(r != DME_RES_OK){
    result = r;
```

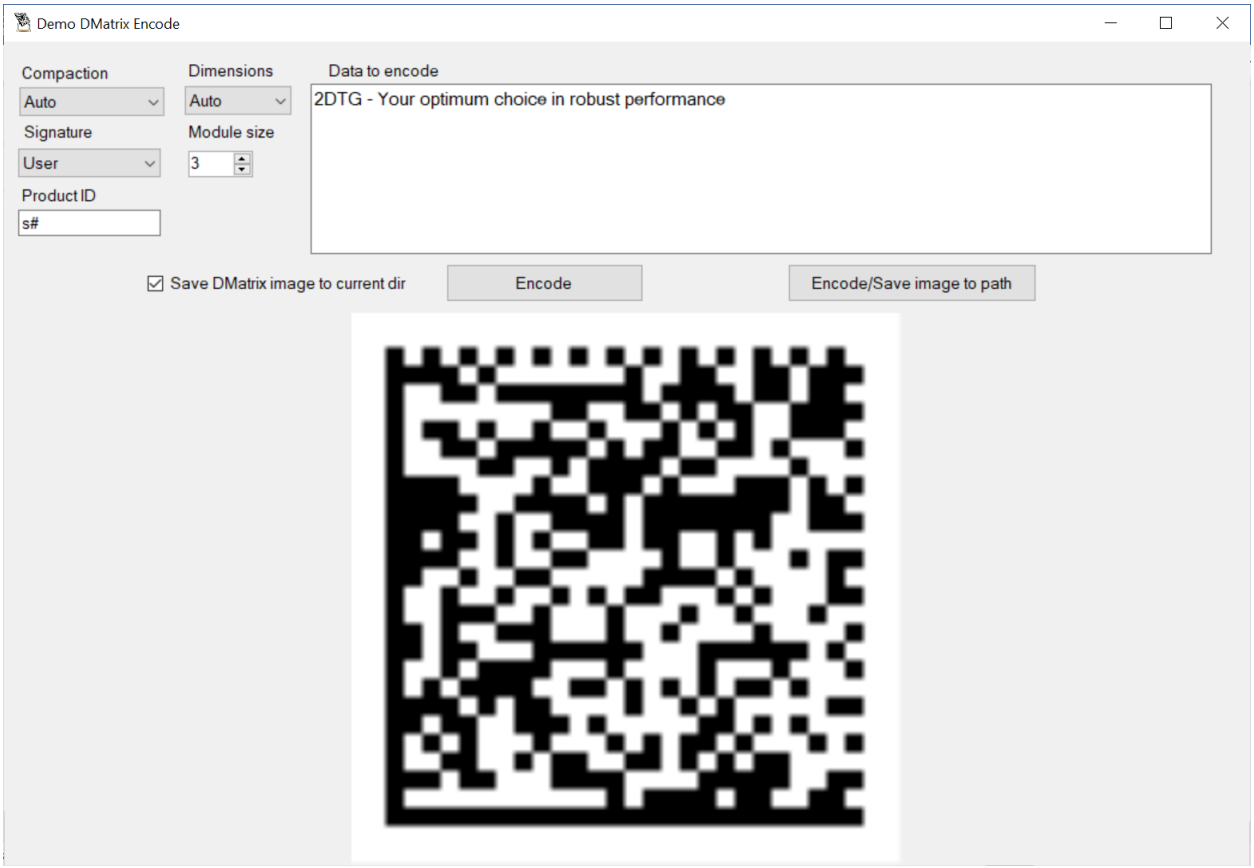
```
      goto exit;
    }
    printf("Greet_u_s1.bmp saved!\n");
  }

exit:
  char ch;
  printf(">");
  scanf("%c", &ch);
  return result;
}
```

4. C# Demo Application GUI

Application is designed to demonstrate full encoding and authentication/encryption functionality of the **DMPS_E** SDK and how to incorporate encoding DLL into the customer application.

Double-click **EncodeDemo.exe** (in Windows 10 you may need to run it as administrator first time you run the program) – **"Demo DMatrix Encode"** window will appear:



Encode Settings Options:

- **Compaction** – specifies whether textual information or a byte array should be used as the barcode's data in its encoding.
- **Dimensions** – specifies Data matrix size.
- **Module size** – specifies Data matrix module size (px) – recommended values: 3-5.

DMPS-Encoding Settings Options:

- **Signature** - choose the option from the drop-down menu:
  - None
  - UserSign – **Authentication** only
  - ProdSign – **Encryption** only
  - Both – **Authentication + Encryption**
- **Product ID** - enter your **Encryption Key** (s#123456789 – evaluation example).

"**Encode**" buttons allow to save symbol either to the DMPS_E folder (default, matrix.bmp) or the folder of your choosing.

Table below depicts examples of different levels of Data Matrix protection:

Protection Level	Data Matrix Type	Data Matrix Sample	Signature Setting
NO Protection	Regular Data Matrix		None
Level "1" Protection **Authentication only**	Authentication enabled Data Matrix		User
Level "2" Protection **Encryption only**	Encrypted Data Matrix		Product
Level "3" Protection **Authentication + Encryption**	Authentication enabled + encrypted Data Matrix		Both