ООО «ЯЛКО»

123557, Москва, ул. Пресненский Вал, Дом 19, строение 1., Тел. +7 916 620-76-04, +7 495 973 8776,
E-mail: managers@yalko.ru, Site: http://www.yalko.ru

Подключкение EMV Kernel к существующему ПО.

Version 2.0 29 ноября 2012 г.

Оглавление

1. Упрощённая схема. Подключение EMV Kernel. 3

1.1. LoadLibrary ("emv_ker.dll"): 4

1.2. Получение адресов функций EMV Kernel library: 4

1.3. Инициализация EMV Kernel: 4

1.4. Создание интерфейсных функций: 4

1.5. Регистрация интерфейсных функций в EMV Kernel 4

1.6. Run EMV Kernel: 5

2. Детальная схема. Подключение EMV Kernel. 5

2.1. Получение адресов функций EMV Kernel library 5

2.1.1. Определение типов указателей на функции. 5

2.1.2. Определение указателей на функции. 5

2.1.3. Загрузка библиотеки. 5

2.1.4. Получение адресов функций. 6

2.1.5. if(hEmvDll != NULL) 6

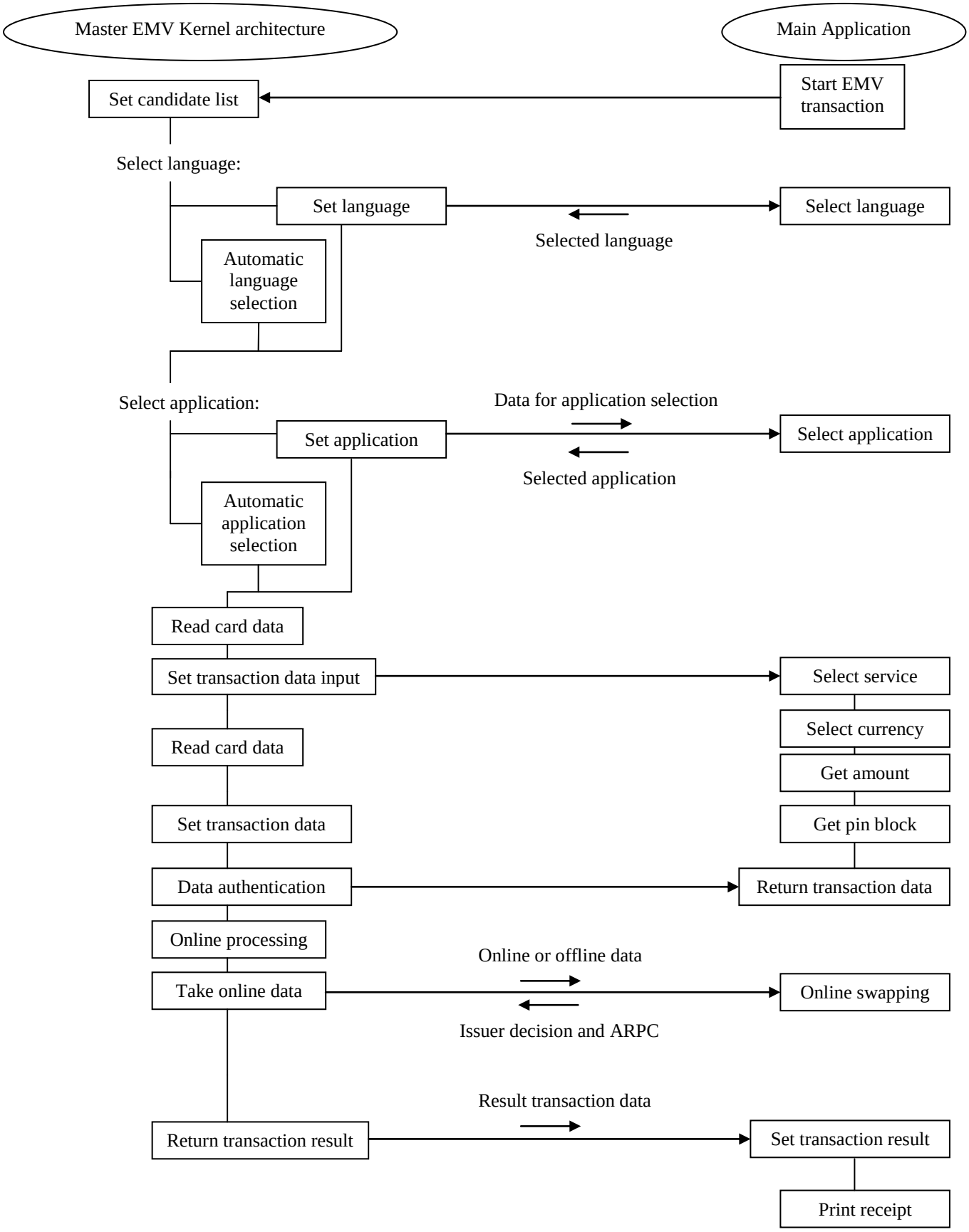
2.2. Регистрация в ядре необходимых внешних функций. 6

2.2.1. Прототипы функций. 6

emv_fAppSelected 7
emv_fSelLang 8
emv_fSelApp 9
emv_fAttGetTrData 10
emv_fGetTrData 11
emv_fGetCryptPin 12
emv_fGetPin 13
emv_fHostSwop 14
emv_fForceAccept 15
emv_fForceOnLine 16
emv_fViewPinRes 17
emv_fIssuerReferral 17
emv_fShowFunds 18

emv_fSelectAccount 19
2.2.2. Определение порядкового номера функций для использования в emv_kernel_registry(). 19
2.2.3. Определение функций. 20
2.2.4. Инициализация ядра. 20
2.2.5. Регистрация внешних функций. 21
2.3. Запуск ядра 21
3. Прототипы функций EMV kernel. 22
emv_kernel_registry 22
emv_kernelInit 23
emv_kernelUnInit 23
emv_kernel_run 24
emv_kernelGetCardData 28
emv_kernelSetCardData 29
emv_kernelSetBerTlvData 30
emv_kernelGetCamData 31
emv_kernelGetErrData 32
emv_kernelClearData 33
emv_kernelGetDataByTag 34
emv_GetTLVBufByDOL 35
emv_GetTLVBufByCDOL 36

1. Упрощённая схема. Подключение EMV Kernel.



1.1. LoadLibrary ("emv_ker.dll"):

1.2. Получение адресов функций EMV Kernel library:

a. GetProcAddress (hEmvDll,"emv_kernel_registry")))
b. GetProcAddress (hEmvDll,"emv_kernelUnInit")))
c. GetProcAddress (hEmvDll,"emv_kernel_run")))
d. GetProcAddress (hEmvDll,"emv_kernelGetCardData")))
e. GetProcAddress (hEmvDll,"emv_kernelSetCardData")))
f. GetProcAddress (hEmvDll,"emv_kernelSetBerTlvData")))
g. GetProcAddress (hEmvDll,"emv_kernelGetCamData")))
h. GetProcAddress (hEmvDll,"emv_kernelGetErrData")))
i. GetProcAddress (hEmvDll,"emv_kernelClearData")))
j. GetProcAddress (hEmvDll,"emv_kernelGetDataByTag")))
k. GetProcAddress (hEmvDll,"emv_GetTLVBufByDOL")))
l. GetProcAddress (hEmvDll,"emv_GetTLVBufByCDOL")))

1.3. Инициализация EMV Kernel:

EmvInitProcAddr(NULL);

1.4. Создание интерфейсных функций:

a. emv_fAppSelected
b. emv_fSelLang
c. emv_fSelApp
d. emv_fAttGetTrData
e. emv_fGetTrData
f. emv_fGetCryptPin
g. emv_fGetPin
h. emv_fHostSwop
i. emv_fForceAccept
j. emv_fForceOnLine
k. emv_fViewPinRes
l. emv_fIssuerReferral
m. emv_fShowFunds
n. emv_fSelectAccount

1.5. Регистрация интерфейсных функций в EMV Kernel

a. EmvRegistryProcAddr((LPVOID)hEmvInit, 1,emv_fAppSelected)
b. EmvRegistryProcAddr((LPVOID)hEmvInit, 2,emv_fSelLang)
c. EmvRegistryProcAddr((LPVOID)hEmvInit, 3,emv_fSelApp)
d. EmvRegistryProcAddr((LPVOID)hEmvInit, 4,emv_fAttGetTrData)
e. EmvRegistryProcAddr((LPVOID)hEmvInit, 5,emv_fGetTrData)
f. EmvRegistryProcAddr((LPVOID)hEmvInit, 6,emv_fGetCryptPin)
g. EmvRegistryProcAddr((LPVOID)hEmvInit, 7,emv_fGetPin)
h. EmvRegistryProcAddr((LPVOID)hEmvInit, 8,emv_fHostSwop)
i. EmvRegistryProcAddr((LPVOID)hEmvInit, 9,emv_fForceAccept)
j. EmvRegistryProcAddr((LPVOID)hEmvInit,10,emv_fForceOnLine)
k. EmvRegistryProcAddr((LPVOID)hEmvInit,11,emv_fViewPinRes)

l. EmvRegistryProcAddr((LPVOID)hEmvInit,12,emv_fIssuerReferral)
m. EmvRegistryProcAddr((LPVOID)hEmvInit,13,emv_fShowFunds)
n. EmvRegistryProcAddr((LPVOID)hEmvInit,14,emv_fSelectAccount)

1.6. Run EMV Kernel:

```
a. TCHAR setup_file[256] = _T("test_ttype14.cfg");
b. TCHAR data[512];
c. DWORD data_len=0;
d. EmvRunProcAddr(hEmvInit, NULL,btTrStatus ,data,&data_len,sizeof(data));
```

2. Детальная схема. Подключение EMV Kernel.

2.1. Получение адресов функций EMV Kernel library.

2.1.1. Определение типов указателей на функции.

```
typedef BYTE  (WINAPI *FUNC_EMV_KERNEL_REGISTRY) (LPVOID
lpHandle,BYTE btfNumber,LPVOID fPointer);
typedef LPVOID(WINAPI *FUNC_EMV_KERNEL_INIT)(char *);
typedef VOID  (WINAPI *FUNC_EMV_KERNEL_UNINIT)(LPVOID lpHandle);
…
```

2.1.2. Определение указателей на функции.

```
FUNC_EMV_KERNEL_REGISTRY        EmvRegistryProcAddr;
FUNC_EMV_KERNEL_INIT            EmvInitProcAddr;
FUNC_EMV_KERNEL_UNINIT          EmvUnInitProcAddr;
…
```

2.1.3. Загрузка библиотеки.

```
hEmvDll = LoadLibrary ("emv_ker.dll");
```

2.1.4. Получение адресов функций.

2.1.5. if(hEmvDll != NULL)

```
{
    If(
      !(
        EmvRegistryProcAddr   = (FUNC_EMV_KERNEL_REGISTRY)
                                GetProcAddress(hEmvDll,"emv_kernel_registry)
      )
    )
    {MessageBox(NULL,"emv_kernel_registry","ERROR",MB_OK);}

    if(!(EmvInitProcAddr          = (FUNC_EMV_KERNEL_INIT)
    GetProcAddress (hEmvDll,"emv_kernelInit")))
    {MessageBox(NULL,"emv_kernelInit","ERROR",MB_OK);}

    if(!(EmvUnInitProcAddr                = (FUNC_EMV_KERNEL_UNINIT          )
    GetProcAddress (hEmvDll,"emv_kernelUnInit")))
    {MessageBox(NULL,"emv_kernelUnInit","ERROR",MB_OK);}
…
}
```

2.2. Регистрация в ядре необходимых внешних функций.

2.2.1. Прототипы функций:

- **emv_fAppSelected**

VOID WINAPI emv_fAppSelected
(
LPVOID ***lpHandle*****,**
LPBYTE ***lpbtLang*****,**
BYTE ***btLangSize*****,**
EMV_APP_INFO * ***lpSelApp***
)

Вызывается из "EMV kernel", сразу после чтения ПАНа карты(таг 5A).

Интерфейсное приложение может запустить ввод пар-ров транзакции в фоновом режиме

Автор:

Konstantin Yalin (C)

Execute:

```
  1.) Получает из EMV kernel ПАН(таг 5A) и трэк(таг 57) исп. функ."emv kernelGetCardData" или
    "emv kernelGetDataByTag".
  2.) Строит список доступных сервисов и операций.
  3.) Запускает процедуру ввода пар-ров транзакции(операция, валюта, сумма), в фоновом
режиме.
```

Use in:

```
      - emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
LPBYTE lpbtLang - pointer to input language buffer
BYTE btLangSize - language buffer size
EMV_APP_INFO *lpSelApp - pointer to selected app data

Возвращает:

void

- **emv_fSelLang**

BYTE WINAPI emv_fSelLang
(
LPVOID ***lpHandle,***
LPBYTE ***lpbtLang,***
BYTE ***btMaxSize***
)

Вызывается из "EMV kernel", в случае необходимости в выборе языка.

.

Предлагает клиенту выбрать язык интерфейса

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует экран выбора языка.
2.) Возвращает в "EMV kernel" язык, выбранный клиентом.
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
LPBYTE lpbtLang - pointer to output language buffer
BYTE btMaxSize - max output buffer size
Exit,: **LPBYTE** lpbtLang - pointer to output language buffer

Возвращает:

BYTE:
0 - canceled
other - language buffer size

- **emv_fSelApp**

BYTE WINAPI emv_fSelApp
(
LPVOID ***lpHandle*****,**
BYTE ***btRetry*****,**
LPBYTE ***lpbtLang*****,**
BYTE ***btLangSize*****,**
EMV_APP_INFO * ***lpAppList*****,**
BYTE ***btAppListSize***
)

Вызывается из "EMV kernel", в случае необходимости в выборе приложения клиентом. Предлагает клиенту выбрать приложение из списка, в соответствии с кандидат-листом

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует экран выбора приложения.
2.) Возвращает в "EMV kernel" порядковый номер, выбранного приложения.
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:

LPVOID	lpHandle	- handle from emv_RunTransaction
BYTE	btRetry	- retry pin entry flag (0 - 1st, other - retry)
LPBYTE	lpbtLang	- pointer to buffer with selected language
BYTE	btLangSize	- buffer with selected language
EMV_APP_INFO	- *lpAppList	- array with app to select info
WORD	wAppListSize	- array size

Возвращает:

WORD:
0xFFFF - cancel
other - selected index

- **emv_fAttGetTrData**

BYTE WINAPI emv_fAttGetTrData
(
LPVOID ***lpHandle,***
LPBYTE ***lpbtLang,***
BYTE ***btLangSize,***
EMV_APP_INFO * ***lpSelApp,***
EMV_TRANS_DATA * ***lpTrData***
)

Вызывается из "EMV kernel", для получения пара-ров транзакции (Операция, Валюта, Сумма).

Using if Attended terminals.

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует экран ввода счёта для перевода средств.(If transaction is transfer)
2.) Формирует экран выбора типа операции.
3.) Формирует экран выбора валюты.
4.) Формирует экран выбора суммы.
5.) Проверяет наличие карты в стоп-листе.
6.) Возвращает в "EMV kernel" выбранные параметры транзакции.
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
LPBYTE lpbtLang - pointer to buffer with selected lang
BYTE btLangSize - buffer with selected lang size
EMV_APP_INFO *lpSelApp - pointer to selected app data
Exit,:
EMV_TRANS_DATA *lpTrData - pointer to buffer to fill with tr data

Возвращает:

BYTE:
0 - ok
other - cancel by Card Holder or Cashier

- **emv_fGetTrData**

BYTE WINAPI emv_fGetTrData
(
LPVOID ***lpHandle,***
EMV_TRANS_DATA * ***lpTrData***
)

Вызывается из "EMV kernel", для получения пара-ров транзакции(Операция, Валюта, Сумма).

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует экран ввода счёта для перевода средств.(If transaction is transfer)
2.) Формирует экран выбора типа операции.
3.) Формирует экран выбора валюты.
4.) Формирует экран выбора суммы.
5.) Проверяет наличие карты в стоп-листе.
6.) Возвращает в "EMV kernel" выбранные параметры транзакции.
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
EMV_APP_INFO *lpSelApp - pointer to selected app data
Exit,: **EMV_TRANS_DATA** *lpTrData - pointer to buffer to fill with tr data

Возвращает:

BYTE:
0 - ok
other - cancel

- **emv_fGetCryptPin**

BYTE WINAPI _EMV_GET_CRYPT_PIN (LPVOID *lpHandle*, LPBYTE *lpbtPinblock*, LPBYTE *lpbtSize*, BYTE *btMaxSize*)

Вызывается из "EMV kernel", в случае необходимости online проверки ПИНа в ЦА.

Предлагает клиенту ввести ПИН карты

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует экран ввода ПИНа.
2.) Возвращает в "EMV kernel" шиврованный ПИН блок.
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:

LPVOID	lpHandle	- handle from emv_RunTransaction

Exit,:

LPBYTE	lpbtPinblock	- pointer to buffer to store entered pin block
LPBYTE	lpbtSize	- pointer to buffer to store entered pin block size
BYTE	btMaxSize	- buffer to store entered pin block size

Возвращает:

BYTE:

0 - ok

1 - timeout (cancel)

other - abort

- **emv_fGetPin**

BYTE WINAPI emv_fGetPin
(
LPVOID ***lpHandle*****,**
BYTE ***btTryCounter*****,**
BYTE ***btRetry*****,**
LPBYTE ***lpbtPinblock*****,**
LPBYTE ***lpbtSize*****,**
BYTE ***btMaxSize***
)

Вызывается из "EMV kernel", в случае необходимости проверки ПИНа картой.
Предлагает клиенту ввести ПИН карты

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует экран ввода ПИНа.
2.) Возвращает в "EMV kernel" введённый ПИН.
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:

LPVOID	lpHandle	- handle from emv_RunTransaction
BYTE	btTryCounter	- PAN Try Counter
BYTE	btRetry	- retry pin entry flag (0 - 1st, other - retry)

Exit,:

LPBYTE	lpbtPinblock	- pointer to buffer to store entered PIN (4-12 ASCII characters)
LPBYTE	lpbtSize	- Length of PIN entered (4-12 ASCII characters)
BYTE	btMaxSize	- buffer to store entered pin block size

Возвращает:

BYTE:
0 - ok
1 - timeout (cancel)
other - abort

- **emv_fHostSwop**

BYTE WINAPI emv_fHostSwop
(
LPVOID ***lpHandle*****,**
LPBYTE ***btAC1*****,**
DWORD ***dwAC1Size*****,**
LPBYTE ***lpbtHostCr*****,**
LPDWORD ***lpdwHostCrSize*****,**
DWORD ***dwMaxSize*****,**
LPBYTE ***lpbtTag71*****,**
LPDWORD ***lpdwTag71Size*****,**
LPBYTE ***lpbtTag72*****,**
LPDWORD ***lpdwTag72Size***
)

Вызывается из "EMV kernel", для связи с ЦА.

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Формирует запрос в соответствии с требованиями эмитента.
1.) Execute связь с ЦА(центром авторизации).
```

Use in:

```
- emv_kernel_run();"EMV kernel"
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
LPBYTE btAC1 - pointer to the buffer with EMV data to send (Created over "DolHostRequest" if present).
Terminal Param file Exsample:
Authoris . DolHostRequest_Len LEN:04
Authoris . DolHostRequest [116]:9F339F35
DWORD dwAC1Size - buffer with EMV data to send size
Exit:
LPBYTE lpbtHostCr - pointer to the buffer to store host EMV data
LPDWORD lpdwHostCrSize - buffer to store host EMV data size
DWORD dwMaxSize - buffer to store host EMV data max length
LPBYTE lpbtTag71 - buffer to store Script 71 data (can be NULL if present into lpbtHostCr)
LPDWORD dwTag71Size - buffer to store Script 71 data Length (can be zero if present into lpbtHostCr)
LPBYTE lpbtTag72 - buffer to store Script 72 data (can be NULL if present into lpbtHostCr)
LPDWORD dwTag72Size - buffer to store Script 72 data Length (can be zero if present into lpbtHostCr)

Возвращает:

BYTE:
0 - ok
other - unable to go on line

- **emv_fForceAccept**

BYTE WINAPI emv_fForceAccept (LPVOID *lpHandle*)

Осуществляет диалог с кассиром если транзакция отклоняется, но кассир имеет право её принять принудительно.

Автор:

Konstantin Yalin (C)

Execute:

```
 1.) Выводит на экран сообщение: Force Accept?.
 2.) Предоставляет меню для выбора Decline или Accept.
```

Use in:

```
     - GenerateAC1();
     - GenerateAC2();
```

Call functions:

```
     - SelectMess();
       Тело функции находится во внешнем приложении, которое использует библиотеку EMV.
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction

Exit,: **None**

Возвращает:

BYTE - Выбор из предложенного меню:
1 => FORCE_ACCEPT_DECLINE;
2 => FORCE_ACCEPT_ACCEPT;
other => FORCE_ACCEPT_DECLINE;

- **emv_fForceOnLine**

BYTE WINAPI emv_fForceOnLine (LPVOID *lpHandle*)

Осуществляет диалог с кассиром если транзакция выполняется в off_line,
но кассир имеет право выполнить её в "On Line" принудительно.

Автор:

Konstantin Yalin (C)

Execute:

```
 1.) Выводит на экран сообщения:
                 - FORCE ON LINE
                 - OFF LINE
                 - ON LINE
 2.) Предоставляет меню для выбора "OFF_LINE" или "ON_LINE".
```

Use in:

```
     - GenerateAC1();
```

Call functions:

```
     - SelectMess();
       Тело функции находится во внешнем приложении, которое использует библиотеку EMV.
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction

Exit,: **None**

Возвращает:

BYTE - Выбор из предложенного меню:
1 => OFF_LINE;
2 => FORCE_ON_LINE;
other => Transaction Decline;

- **emv_fViewPinRes**

VOID WINAPI emv_fViewPinRes (LPVOID *lpHandle*, BYTE *btPinResult*)

Выводит на экран результат проверки Plaintext(off_line) PIN.

Автор:

Konstantin Yalin (C)

Execute:

```
  1.) Выводит на экран сообщение "INCORRECT PIN" or "PIN OK":
```

Аргументы:

Entry,:
BYTE btPinResult;
1 => PIN_OK
other => INCORRECT_PIN
Exit,: **None**

Возвращает:

void

- **emv_fIssuerReferral**

BYTE WINAPI emv_fIssuerReferral (LPVOID *lpHandle*, LPBYTE *lpbtPan*, BYTE *btSize*)

When the Authorization Response Code in the authorization response message indicates **that** a voice referral should be performed by the attendant, prior to issuing the second GENERATE AC command, **an** attended terminal shall either display the 'CALL YOUR BANK' message to the attendant, **or** shall alert the attendant in some other way that the Issuer has requested a voice referral.

Автор:

Konstantin Yalin (C)

Execute:

```
  1.) Выводит на экран сообщение "CALL YOUR BANK"(сообщает кассиру о необходимости связи с
бБанком).
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
BYTE lpbtPan - pointer to PAN **BYTE** btSize - Lenrth PAN Exit: **None**

Возвращает:

BYTE result:
1 => Transaction Approved (TC).
0 => Transaction Decline (AAC).

- **emv_fShowFunds**

VOID WINAPI emv_fShowFunds
(
LPVOID ***lpHandle,***
LPBYTE ***lpbtFunds,***
BYTE ***btSize***
)

The terminal shall display the VLP Available Funds after reading the response to the Second GENERATE AC command.

Terminal may display or print VLP Available Funds if function is supported. The terminal shall display the VLP Available Funds anytime after it is read during the Read Application Date phase. Konstantin Yalin (C)

Execute:

```
1.) Выводит на экран сообщение о размере фондов карты клиента.
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
LPBYTE lpbtFunds - Pointer to Funds: tag 9F79[6];
BYTE btSize - Lenпth of Funds: sizeof(tag 9F79);

Exit: **None**

Возвращает:

void

- **emv_fSelectAccount**

BYTE WINAPI emv_fSelectAccount
(
LPVOID ***lpHandle*****,**
LPBYTE ***lpbtLang*****,**
BYTE ***btLangSize*****,**
EMV_APP_INFO * ***lpSelApp***
)

Осуществляет диалог с клиентом, если требуется выбор счёта.

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Выводит на экран сообщения:
                - Select Account:
                - Savings
                - Cheque/debit
                - Credit
2.) Предоставляет меню для выбора.
```

Use in:

```
- EmvStart();
```

Call functions:

```
- SelectMess();
  Тело функции находится во внешнем приложении, которое использует библиотеку EMV.
```

Аргументы:

Entry,:
LPVOID lpHandle - handle from emv_RunTransaction
LPBYTE lpbtLang - pointer to input language buffer
BYTE btLangSize - language buffer size
EMV_APP_INFO *lpSelApp - pointer to selected app data
Exit,: **None**

Возвращает:

BYTE - Выбор из предложенного меню:
0x00 => Default – no account specified.
0x10 => Savings
0x20 => Cheque/debit
0x30 => Credit
0xFF => Terminate Transaction

2.2.2. Определение порядкового номера функций для использования в emv_kernel_registry().

```
#define NUM_FUNC_EMV_APP_SELECTED        1
#define NUM_FUNC_EMV_SELECT_LANG         2
#define NUM_FUNC_EMV_SELECT_APP          3
#define NUM_FUNC_EMV_ATT_GET_TR_DATA     4
```

```
#define NUM_FUNC_EMV_GET_TR_DATA          5
#define NUM_FUNC_EMV_GET_CRYPT_PIN        6
#define NUM_FUNC_EMV_GET_PIN              7
#define NUM_FUNC_EMV_HOST_SWOP            8
#define NUM_FUNC_EMV_FORCE_ACCEPT         9
#define NUM_FUNC_EMV_FORCE_ON_LINE       10
#define NUM_FUNC_EMV_PIN_ENTRY_RESULT    11
#define NUM_FUNC_EMV_ISSUER_REFERRAL     12
#define NUM_FUNC_EMV_SHOW_FUNDS          13
#define NUM_FUNC_EMV_SELECT_ACCOUNT      14
```

2.2.3. Определение функций.

```
void WINAPI emv_fAppSelected(LPVOID lpHandle, LPBYTE lpbtLang, BYTE
btLangSize, emv_APP_INFO *lpSelApp)
{
    …
}
BYTE WINAPI emv_fSelLang(LPVOID lpHandle, LPBYTE lpbtLang, BYTE btMaxSize)
{
    …
}
BYTE WINAPI emv_fSelApp(LPVOID lpHandle, BYTE btRetry, LPBYTE lpbtLang,
BYTE btLangSize, emv_APP_INFO *lpAppList, BYTE btAppListSize)
{
    …
}
...
```

2.2.4. Инициализация ядра.

```
if(EmvInitProcAddr)
    {
        hEmvInit = (LONG)EmvInitProcAddr(NULL);
        if(!hEmvInit){
            MessageBox(NULL,"emv_kernelInit","EmvWin ERROR!",MB_OK);
        }
    }
```

2.2.5. Регистрация внешних функций.

```
if(EmvRegistryProcAddr)
{
If (
    EmvRegistryProcAddr(
            (LPVOID)hEmvInit,
            NUM_FUNC_EMV_APP_SELECTED,
            emv_fAppSelected)
)MessageBox(NULL,_T("emv_fAppSelected"),_T("err"),MB_OK);

if(
    EmvRegistryProcAddr(
            (LPVOID)hEmvInit,
            NUM_FUNC_EMV_SELECT_LANG,
            emv_fSelLang)
) MessageBox(NULL,_T("emv_fSelLang"),_T("err"),MB_OK);
if(
    EmvRegistryProcAddr(
            (LPVOID)hEmvInit,
            NUM_FUNC_ EMV_SELECT_APP, emv_fSelApp)
) MessageBox(NULL,_T("emv_fSelApp"),_T("err"),MB_OK);
…
}
```

2.3. Запуск ядра.

```
DWORD rc_dword;
rc_dword=(BYTE)EmvRunProcAddr(
            (LPVOID)hEmvInit,
            NULL, btTrStatus ,
            (BYTE*)data,
            &data_len,
            sizeof(data));
```

Структуру ядра можно посмотреть на сайте www.yalko.ru

3. Прототипы функций EMV kernel:

- **emv_kernel_registry**

BYTE WINAPI emv_kernel_registry
(
LPVOID ***lpHandle,***
BYTE ***btfNumber,***
LPVOID ***fPointer***
)

Initialization interface function pointers.

Автор:

Konstantin Yalin (C)

Аргументы:

Entry,:
LPVOID lpHandle - to pass as param to callback
BYTE btfNumber - interface function sequence number.(look: define NUM_FUNC_EMV_ ...)
LPVOID fPointer - applicable function pointer.

Возвращает:

BYTE:
0 - ok
other - error

- **emv_kernelInit**

LPVOID WINAPI emv_kernelInit (char * *InterfaceDllname*)

Initialization EMV DLL.

Автор:

: Konstantin Yalin (C)

Execute:

```
 1.) Инициализация объектов даных EMV транзакции.
 2.) Чтение параметров EMV из файла "emvsetup.txt".
 3.) Загрузка библиотеки "SmartCard.dll":
```

Use in:

```
      - В приложениях, использующих "EMV kernel".
```

In **********

Аргументы:

Entry,:
char * InterfaceDllName - Interface functions DLL name. Need if you use

Возвращает:

NULL - failed
other - handle to pass to other functions

- **emv_kernelUnInit**

VOID WINAPI emv_kernelUnInit (LPVOID *lpHandle*)

De-Initialization EMV DLL.

Автор:

: Konstantin Yalin (C)

Execute:

```
 1.) Close smart-reader.
 3.) Reload library "SmartCard.dll".(FreeLibrary (hSmarCardDll);)
```

Use in:

```
      - В приложениях, использующих "EMV kernel".
```

In **********

Аргументы:

Entry,:
LPVOID lpHandle - to pass as param to callback

Возвращает:

VOID

- **emv_kernel_run**

DWORD WINAPI emv_kernel_run
(
LPVOID lpHandle,
LPVOID lpCallbackHandle,
BYTE btTrStatus,
LPBYTE lpbtCr,
LPDWORD lpdwCrSize,
DWORD dwMaxSize
)

Assignment:
Execute transaction EMV card in conformity
EMV Integrated Circuit Card Specifications for Payment Systems Version 4.1 May 2004

Автор:

: Konstantin Yalin (C)

Execute:

```
1.) Посылает на карту команду Answer To Reset (ATR).
2.) Выбирает предпочтительный язык интефейса клиента.
3.) Выбирает приложение для проведения транзакции.
4.) Получает параметры транзакции(Тип операции,Сумма,Валюта).
5.) Производит чтенние данных с карты клиента.
6.) Получает ПИН клиента.
7.) Передаёт данные в ЦА.
8.) Получает ответ из ЦА.
```

Call functions:

```
- EMV SELECT LANG       emv fSelLang();
- EMV_SELECT_APP        emv_fSelApp();
- EMV_GET_TR_DATA       emv_fGetTrData();
- EMV_GET_CRYPT_PIN     emv_fGetCryptPin();
- EMV GET PIN           emv fGetPin();
- EMV HOST SWOP         emv fHostSwop();
- ...
```

Use in:

```
- В приложениях, использующих "EMV kernel".
```

Аргументы:

Entry,:
LPVOID lpHandle - to pass as param to callback
LPVOID lpCallbackHandle - to pass as param to callback
BYTE btTrStatus - Transaction Status:
0 => New card.
1 => The same card, the same Application .
2 => The same card, new select Application
LPBYTE lpbtCr - Указатель на данные транзакции в формате TLV.
LPDWORD lpdwCrSize - Указатель на кол-во данных.
DWORD dwMaxSize - Размер буфера для данных.
Exit,:
LPBYTE lpbtCr - pointer to the BER TLV buffer with EMV transaction data.
(Created over "DolHostReversal or DolHostAdvice" if present in terminal param file)
Terminal Param file Exsample:

Authoris. DolHostReversal_Len LEN:04
Authoris. DolHostReversal [116]:9F339F35
Authoris. DolHostAdvice_Len LEN:04
Authoris. DolHostAdvice [116]:9F339F35
LPDWORD lpdwCrSize - Указатель на кол-во данных.

Возвращает:

BYTE:
AC1:
0x00 - approved
0x01 - force accept
0x02 - decline
0x03 - abort
0x04 - capture
0x05 - decline & advice
AC2:
0x10 - approved
0x11 - force accept
0x12 - decline
0x13 - reversal
0x14 - abort + reversal
0x15 - capture
0x16 - capture & reversal
0x17 - approved + advice
0x18 - force accept + advice
0x19 - decline + advice
AC2 + Script Result is Ready:
0x90 - approved
0x91 - force accept
0x92 - decline
0x93 - reversal
0x94 - abort + reversal
0x95 - capture
0x96 - capture & reversal
0x97 - approved + advice
0x98 - force accept + advice
0x99 - decline + advice
Other:
0x20 - Transaction without GAC 2 approved
0x22 - Transaction without GAC 2 declined
0xA0 - Transaction without GAC 2 approved + Script Results Ready
0xA2 - Transaction without GAC 2 declined + Script Results Ready
0x55 - Not Accepted
0x66 - Card Error.

Returned Value	Off Line Only	Off line + On Line	On Line Only	Description
0x00 - approved	+	+	-	Транзакция завершилась успешно без участия ЦА, после GAC1
0x01 - force accept	+	-	-	Транзакция завершилась успешно «принудительно», по решению кассира.

0x02 - decline	+	+	+	Транзакция завершилась отказом без участия ЦА, после GAC1
0x03 - abort	+	+	+	Транзакция прервана. Причина может быть выяснена после рассмотрения подробного «трейса».
0x04 - capture	+	+	+	Карта подлежит изъятию.
0x05 - decline & advice	+	+	+	Транзакция завершилась отказом без участия ЦА, после GAC1. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций.
0x10 - approved	-	+	+	Транзакция завершилась успешно с участием ЦА, после GAC2.
0x11 - force accept	-	+	-	Транзакция завершилась успешно «принудительно», по решению кассира и с участием ЦА.
0x12 - decline	-	+	+	Транзакция завершилась отказом с участием ЦА.
0x13 - reversal	-	+	+	Транзакция завершилась отказом с участием ЦА, после GAC2. Терминал должен провести отмену.
0x14 - abort + reversal	-	+	+	Транзакция завершилась отказом, по причине ошибки карты и после обмена данными с ЦА. Терминал должен провести отмену.
0x15 - capture	-	+	+	Транзакция завершилась отказом с участием ЦА. Карта подлежит изъятию.
0x16 - capture & reversal	-	+	+	Транзакция завершилась отказом с участием ЦА, после GAC2. Терминал должен провести отмену. Карта подлежит изъятию.
0x17 - approved + advice	-	+	+	Транзакция завершилась успешно с участием ЦА, после GAC2. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций.
0x18 - force accept + advice	-	+	-	Транзакция завершилась успешно «принудительно», по решению кассира и с участием ЦА. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций.
0x19 - decline + advice	-	+	+	Транзакция завершилась отказом с участием ЦА. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций.
0x90 - approved	-	+	+	Транзакция завершилась успешно с участием ЦА, после GAC2. Выполнены «скрипты» присланные ЦА.
0x91 - force accept	-	+	+	Транзакция завершилась успешно «принудительно», по решению кассира и с участием ЦА. Выполнены «скрипты» присланные ЦА.
0x92 - decline	-	+	+	Транзакция завершилась отказом с участием ЦА. Выполнены «скрипты» присланные ЦА.
0x93 - reversal	-	+	+	Транзакция завершилась отказом с участием ЦА, после GAC2. Терминал должен провести отмену.

				Выполнены «скрипты» присланные ЦА.
0x94 - abort + reversal	-	+	+	Транзакция завершилась отказом, по причине ошибки карты и после обмена данными с ЦА. Терминал должен провести отмену. Выполнены «скрипты» присланные ЦА.
0x95 - capture	-	+	+	Транзакция завершилась отказом с участием ЦА. Карта подлежит изъятию. Выполнены «скрипты» присланные ЦА.
0x96 - capture & reversal	-	+	+	Транзакция завершилась отказом с участием ЦА, после GAC2. Терминал должен провести отмену. Карта подлежит изъятию. Выполнены «скрипты» присланные ЦА.
0x97 - approved + advice	-	+	+	Транзакция завершилась успешно с участием ЦА, после GAC2. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций. Выполнены «скрипты» присланные ЦА.
0x98 - force accept + advice	-	+	+	Транзакция завершилась успешно «принудительно», по решению кассира и с участием ЦА. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций. Выполнены «скрипты» присланные ЦА.
0x99 - decline + advice	-	+	+	Транзакция завершилась отказом с участием ЦА. Терминал должен послать соответствующее сообщение в ЦА или сформировать соответствующую запись в журнале транзакций. Выполнены «скрипты» присланные ЦА.
0x20	-	+	+	Transaction without GAC 2 approved. Используется для ЦА поддерживающих транзакции без GAC2.
0x22	-	+	+	Transaction without GAC 2 declined. Используется для ЦА поддерживающих транзакции без GAC2.
0xA0	-	+	+	Transaction without GAC 2 approved + Script Results Ready. Используется для ЦА поддерживающих транзакции без GAC2.
0xA2	-	+	+	Transaction without GAC 2 declined + Script Results Ready. Используется для ЦА поддерживающих транзакции без GAC2.
0x55 - Not Accepted	+	+	+	Карта не обслуживается, терминал может попробовать обслужит карту по магнитной полосе, в зависимости от условий определяемых платежной системой.
0x66 - Card Error.	+	+	+	Карта не обслуживается, терминал может попробовать обслужит карту по магнитной полосе, в зависимости от условий определяемых платежной системой.

- **emv_kernelGetCardData**

DWORD WINAPI emv_kernelGetCardData
(
WORD *dwTag*,
LPBYTE *lpbtData*,
LPDWORD *lpdwSize*,
DWORD *dwMaxSize*
)

Возвращает объекты данных, EMV транзакции.

Assignment:

Автор:

: Konstantin Yalin (C)

Execute:

```
1.) Возвращает объекты данных, EMV транзакции.
```

Use in:

```
    - В приложениях, использующих "EMV kernel".
```

Аргументы:

Entry,:

WORD	dwTag	- requested tag (sample: 0x9F02)
LPBYTE	lpbtData	- output buffer
LPDWORD	lpdwSize	- output buffer length
DWORD	dwMaxSize	- max output buffer size

Exit,:

LPBYTE	lpbtData	- output buffer
LPDWORD	lpdwSize	- output buffer length

Возвращает:

BYTE:
0 - ok
other - error

- **emv_kernelSetCardData**

DWORD WINAPI emv_kernelSetCardData
(
WORD ***dwTag,***
LPBYTE ***lpbtData,***
DWORD ***dwSize***
)

Set data Emv Application according to specified tag.

Автор:

: Konstantin Yalin (C)

Execute:

```
1.) Устанавливает объекты данных, EMV транзакции.
```

Use in:

```
    - В приложениях, использующих "EMV kernel".
```

Аргументы:

Entry,:
WORD dwTag - requested tag (sample: 0x9F41)
LPBYTE lpbtData - input buffer
LPDWORD dwSize - input buffer length
Exit,: **None**

Возвращает:

DWORD:
0 - ok
other - error

- **emv_kernelSetBerTlvData**

BYTE WINAPI emv_kernelSetBerTlvData
(
LPVOID ***lpHandle,***
LPBYTE ***lpbtBuffer,***
DWORD ***dwBufferSize***
)

Set EMV object data.

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Set EMV object data
```

Аргументы:

Entry,:
LPVOID lpHandle - from emv_kernelInit
LPBYTE lpbtBuffer - buffer with tlv objects
DWORD dwBufferSize - buffer size
Exit,: **None**

Возвращает:

BYTE:
0 - ok
other - error

- **emv_kernelGetCamData**

BYTE WINAPI emv_kernelGetCamData
(
LPVOID ***lpHandle,***
LPBYTE ***lpbtByte1,***
LPBYTE ***lpbtByte2***
)

Get CAM Data (NDC specific).

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Get CAM Data (NDC specific)
```

Аргументы:

Entry,:
LPVOID lpHandle - from emv_kernelInit
LPBYTE lpbtByte1 - buffer with byte1
LPBYTE lpbtByte2 - buffer with byte2
Exit,: **None**

Возвращает:

BYTE:
0 - ok
other - error

- **emv_kernelGetErrData**

VOID WINAPI emv_kernelGetErrData
(
LPVOID ***lpHandle,***
LPBYTE ***lpbtByte1,***
LPBYTE ***lpbtByte2***
)

Set bits mask for errors.

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Set bits mask for errors.
```

Аргументы:

Entry,:
LPVOID lpHandle - LPVOID lpHandle - from scs_emv_kernelInit
LPBYTE lpbtByte1 - buffer with byte1
LPBYTE lpbtByte2 - buffer with byte2
Exit,:

```
Bit Value Description
Byte 1:
8 - The card was blocked, and therefore no application could be selected.
7 - The candidate list was empty because the MasterCard application was blocked
6 - Candidate List is empty.
5 - Card Rider have a technical problem. (Get ATR returned error).
4 - Inconsistency between PAN (Tag5A) and  Track 2  (Tag 57).
3 - Reserved.
2 - Reserved.
1 - Reserved.

Byte 2:
8 - Reserved.
7 - Reserved.
6 - Reserved.
5 - Reserved.
4 - Reserved.
3 - Reserved.
2 - Reserved.
1 - Reserved.
```

Возвращает:

void

- **emv_kernelClearData**

API BYTE WINAPI emv_kernelClearData (LPVOID *lpHandle*)

Clears all transaction-related data.

Автор:

Konstantin Yalin (C)

Execute:

```
        1.) Clears all transaction-related data
```

Аргументы:

Entry,:
LPVOID lpHandle - from emv_kernelInit
Exit,:
None

Возвращает:

BYTE:
0 - ok
other - error

- **emv_kernelGetDataByTag**

BYTE WINAPI emv_kernelGetDataByTag
(
LPVOID ***lpHandle*****,**
LPBYTE ***lpbtTag*****,**
DWORD ***dwTagSize*****,**
LPBYTE ***lpbtData*****,**
LPDWORD ***lpdwSize*****,**
DWORD ***dwMaxSize***
)

Get Data from card by TAG.

Автор:

Konstantin Yalin (C)

Execute:

```
1.) Get Data from card by TAG
```

Аргументы:

Entry,:

LPVOID	lpHandle	- from emv_kernelInit
LPBYTE	lpbtTag	- buffer with requested TAG
DWORD	dwTagSize	- buffer with requested TAG size
DWORD	dwMaxSize	- max output buffer limit

Exit,:

LPBYTE	lpbtData	- output buffer
LPDWORD	lpdwSize	- output buffer size

Возвращает:

BYTE:
0 - ok
other - error

- **emv_GetTLVBufByDOL**

BYTE WINAPI EMV_GetTLVBufByDOL
(
LPVOID ***lpHandle,***
LPBYTE ***dol,***
DWORD ***dol_size,***
LPBYTE ***out,***
DWORD * ***out_len,***
DWORD ***out_max***
)

Функция в соответствии с ДОЛ формирует блок данных в формате.
(TLV: TAG, LENGTH, VALUE).

Автор:

Konstantin Yalin (C)

Execute:

```
  1.) Функция в соответствии с ДОЛ формирует данные в формате (TLV: TAG,LENGTH,VALUE)
Например:
dol  =  9F279F369F26BF10
data = {0x80,0x00,0x06,0xA6,0x3B,0xAB,0xE9,0x01};
=>
TAG 9F27=80
TAG 9F36=0006
TAG 9F26=A63BABE9
TAG BF10=01
out=9F270180 9F36020006 9F2604A63BABE9 BF100101
```

In **********

Аргументы:

Entry,:
LPVOID lpHandle - to pass as param to callback
LPBYTE dol; - ДОЛ, в соответствии с которым группируются данные.
DWORD dol_size; - Размер ДОЛа.
DWORD out_max; - max output buffer size

Exit,:
LPBYTE out; - Указатель на буфер размещения сгруппированных данных.
DWORD *out_len - output buffer length.

Возвращает:

BYTE - status **0** => ok **other** => error

- **emv_GetTLVBufByCDOL**

BYTE WINAPI EMV_GetTLVBufByCDOL
(
LPVOID ***lpHandle,***
BYTE ***flag_cdol,***
LPBYTE ***lpbtBuffer,***
DWORD ***dwBufferSize,***
LPBYTE ***out,***
DWORD * ***out_len,***
DWORD ***out_max***
)

Функция в соответствии с ДОЛ формирует блок данных в формате.
(TLV: TAG, LENGTH, VALUE).

Автор:

Konstantin Yalin (C)

Execute:

```
  1.) Формирует сообщение по транзакции в соответствии BOOK-4(8.1 Message Content).
      - Если установлен флаг (flag cdol == CDOL1 || == CDOL2)
        формирует данные в (LPBYTE out), в формате TLV на основании CDOL.
  2.) Функция в соответствии с списком тагов(LPBYTE lpbtBuffer), добавляет к (LPBYTE
out), данные в формате (TLV: TAG,LENGTH,VALUE)
      Например:
      dol  =  9F279F369F26BF10
      data = {0x80,0x00,0x06,0xA6,0x3B,0xAB,0xE9,0x01};
      =>
      TAG 9F27=80
      TAG 9F36=0006
      TAG 9F26=A63BABE9
      TAG BF10=01
      out=9F270180 9F36020006 9F2604A63BABE9 BF100101
```

In **********

Аргументы:

Entry,:
LPVOID lpHandle - to pass as param to callback
BYTE flag_cdol 1 => CDOL1(tag 8C), 2 => CDOL2(tag 8D)
LPBYTE dol - ДОЛ, в соответствии с которым группируются данные.
DWORD dol_size - Размер ДОЛа.
DWORD out_max - max output buffer size

Exit,:
LPBYTE out - Указатель на буфер размещения сгруппированных данных.
DWORD *out_len - output buffer length.

Возвращает:

BYTE - status **0** => ok **other** => error

4. Использование считывателей. (smart-card readers)

Для передачи данных между картой и ядром используется набор функций и методов, которые должны быть размещены в динамически присоединяемой библиотеке “SmartCard.dll”.

В процессе транзакции, Ядро (emv_ker.dll) вызывает следующие методы:

```
WORD WINAPI SendApduIn(BYTE *Cmd,BYTE * Data,WORD *Data_len,
                       BYTE*answer,WORD answer_max_len);
BYTE WINAPI Atr (BYTE *atr, DWORD * max_len,BYTE * protocol);
WORD WINAPI SendApduOut(BYTE *Cmd,WORD *Stat,BYTE *answer,
                        WORD answer_max_len);
BYTE WINAPI CheckConnect(void);
```

Примеры использование методов **Atr, CheckConnect, SendApduIn, SendApduOut,** в “Emv_Ker.dll:

```
{
        BYTE answer[300];
        WORD stat=0;
        BYTE cmd[5];
        BYTE data[300];
        WORD data_len;

        stat = SendApduIn(cmd, data, &data_len, answer, sizeof(answer));
        data_len = SendApduOut(cmd, &stat, answer, sizeof(answer));

}
{
        BYTE atr[256];
        DWORD atr_len=sizeof(atr);
        BYTE protocol;
        AnswerToReset(atr,&atr_len,&protocol))
}

{
        BYTE rc = AddrCheckConnect ();
        if(rc)   return 1;// TRUE
        else return 0;// FALSE
}
```

4.1. USB readers:

4.2. Моторизованные считыватели:

Спецификация
на SmartCard.dll для управления моторизованным ридером.

Методы:

1) **OpenReader()**
2) **CheckConnect()**
3) **Atr()**
4) **SendApduIn()**
5) **SendApduOut()**
6) **TakeCard ()**
7) **GetCardBack()**
8) **CloseReader()**

Описание методов:

BYTE WINAPI OpenReader(LPSTR rname);
Инициализирует и открывает ридер.
Параметры:
rname - имя COM порта. (“COM1”)
Возвращаемые значения:
0 –успешно, иначе – ошибка при открытии.

BYTE WINAPI CheckConnect();
Вызывается emv_kernel.dll
Проверяет наличие карточки в ридере.
Возвращаемые значения:
0-карта есть в ридере, инче – карты нет в ридере.

BYTE WINAPI Atr (BYTE *atr, DWORD * max_len,BYTE * protocol) ;
Вызывается emv_kernel.dll
Получает с карточки Atr и передаёт ядру.
Параметры:
atr – указатель на массив байтов.
max_len – указатель на значение длинны массива. По этому указателю в emv_kernel.dll передаётся длинна atr.
protocol – указатель на значение протокола. По этому указателю в emv_kernel.dll передаётся протокол, используемый карточкой.
Возвращаемые значения:
1 – успешно, 0 – не удачно.

WORD WINAPI SendApduIn(BYTE *Cmd,BYTE * Data,WORD *Data_len,BYTE *answer,WORD answer_max_len)
Вызывается emv_kernel.dll
Посылает Apdu команду на карточку.
Параметры:
Cmd – указатель на массив, содержащий команду, длинна массива 5 байт.
Data – указатель на массив, содержащий данные.

Data_len – размер данных.
answer – указатель на массив для ответа от карточки.
answer_max_len – длинна массива для ответа от карточки.
Возвращаемые значения:
Возвращает статус выполнения команды.

WORD WINAPI SendApduOut(BYTE *Cmd,WORD *Stat,BYTE *answer,WORD answer_max_len)
Вызывается emv_kernel.dll
Посылает Apdu команду на карточку.
Параметры:
Cmd – указатель на массив, содержащий команду, длинна массива 5 байт.
Stat – указатель на значение для статуса выполнения команды.
answer – указатель на массив для ответа от карточки.
answer_max_len – длинна массива для ответа от карточки.
Возвращаемые значения:
длинна ответа от карточки.

BYTE WINAPI TakeCard ()
Посылает ридеру команду принять карту.
Возвращаемые значения:
0 – успешно, иначе – не удачно.

BYTE WINAPI GetCardBack()
Возвращает карточку
Возвращаемые значения:
0 – успешно, иначе – не удачно.

BYTE WINAPI CloseReader()
Закрывает ридер и отключает его.
Возвращаемые значения:
0 – успешно, иначе – не удачно.

Перед вызовом emv_kernell_run() нужно открыть ридер.
Закрывать его надо после вызова emv_kernel_uninit()

4.2.1. ICT 3K5:

4.2.2. NEURON:

```
BYTE WINAPI OpenReader(LPSTR comPort);
```

Описание:
Открывает COM порт и инициализирует ридер. Если COM порт уже открыт, производится только инициализация ридера.

Параметры:
comPort – имя COM порта.

Возвращаемые значения:
0 – операция завершилась успешно.

1 – ошибка записи/чтения.
2 – несоответствие размера буфера данных и количества байтов для записи/чтения.
3 – COM порт не был открыт.
4 – ошибка создания потоков записи/чтения.
5 – был получен NAK от ридера, что может означать, что не правильно посчитано LRC.
6 – ридер вернул плохое LRC.
7 – ошибка открытия COM порта.
8 – команда не может быть выполнена.

0x31 – карта держится внутри ридера.
0x32 – ошибка открытия/закрытия шторки.
0x33 – ошибка транспортировки карты.
0x34 – ошибка подачи карты.

BYTE WINAPI CheckConnect();

Описание:
Проверяет наличие контакта с чипом. Функция используется EMV ядром.

Возвращаемые значения:
1 – есть контакт
0 – нет контакта.

BYTE WINAPI EjectCard();

Описане:
Возвращает карту клиенту.

Возвращаемые значения:
0 – операция завершилась успешно.
1 – ошибка записи/чтения.
2 – несоответствие размера буфера данных и количества байтов для записи/чтения.
3 – COM порт не был открыт.
4 – ошибка создания потоков записи/чтения.
5 – был получен NAK от ридера, что может означать, что не правильно посчитано LRC.
6 – ридер вернул плохое LRC.

0x31 – карта держится внутри ридера.
0x32 – ошибка открытия/закрытия шторки.
0x33 – ошибка транспортировки карты.
0x34 – ошибка подачи карты.

BYTE WINAPI GetCardOut();

Описание:

Сбрасывает карту в контейнер.

Возвращаемые значения:
0 – операция завершилась успешно.
1 – ошибка записи/чтения.
2 – несоответствие размера буфера данных и количества байтов для записи/чтения.
3 – COM порт не был открыт.
4 – ошибка создания потоков записи/чтения.
5 – был получен NAK от ридера, что может означать, что не правильно посчитано LRC.
6 – ридер вернул плохое LRC.

0x31 – карта держится внутри ридера.
0x32 – ошибка открытия/закрытия шторки.
0x33 – ошибка транспортировки карты.
0x34 – ошибка подачи карты.

BYTE WINAPI TakeCard(BYTE * buff, DWORD * buffSize);

Описание:
Ожидает подачи карты. После подачи считывает все 3 трека магнитной полосы и возвращает в буфер. На основе данных полосы определяет, возможна транзакция по чипу или нет.

Параметры:
BYTE * buff – буфер для приёма значения магнитной полосы.
DWORD * buffSize – размер буфера для приёма значения магнитной полосы. После завершения работы функции сюда передаётся размер считанных данных.

Возвращаемые значения:
0 – операция завершилась успешно, возможна работа с чипом.
0x0A - операция завершилась успешно, чип не доступен.
1 – ошибка записи/чтения.
2 – несоответствие размера буфера данных и количества байтов для записи/чтения.
3 – COM порт не был открыт.
4 – ошибка создания потоков записи/чтения.
5 – был получен NAK от ридера, что может означать, что не правильно посчитано LRC.
6 – ридер вернул плохое LRC.
9 – не было подачи карты.

0x31 – карта держится внутри ридера.
0x32 – ошибка открытия/закрытия шторки.
0x33 – ошибка транспортировки карты.
0x34 – ошибка подачи карты.

BYTE WINAPI CloseReader();

Описание:
Прекращает работу ридера и закрывает COM порт. В случае, если в ридере находится карта, возвращает её клиенту.

Возвращаемые значения:
0 – операция завершилась успешно.
1 – ошибка записи/чтения.
2 – несоответствие размера буфера данных и количества байтов для записи/чтения.
3 – COM порт не был открыт.
4 – ошибка создания потоков записи/чтения.
5 – был получен NAK от ридера, что может означать, что не правильно посчитано LRC.
6 – ридер вернул плохое LRC.

0x31 – карта держится внутри ридера.
0x32 – ошибка открытия/закрытия шторки.
0x33 – ошибка транспортировки карты.
0x34 – ошибка подачи карты.

BYTE WINAPI DisConnect();

Описание:
Отключает питание от чипа, возвращает карту и удерживает её на выходе.

Возвращаемые значения:
0 – операция завершилась успешно.
1 – ошибка записи/чтения.
2 – несоответствие размера буфера данных и количества байтов для записи/чтения.
3 – COM порт не был открыт.
4 – ошибка создания потоков записи/чтения.
5 – был получен NAK от ридера, что может означать, что не правильно посчитано LRC.
6 – ридер вернул плохое LRC.

BYTE WINAPI Atr(BYTE *atr, DWORD * max_len, BYTE * protocol);

Описание:
Подаёт питание на чип и возвращает ответ (ATR).

Параметры:
BYTE * atr – буфер для приёма значения ATR.
DWORD * max_len – размер буфера для приёма ATR, возвращается размер данных ATR.
BYTE * protocol – всегда возвращает 0. Используется EMV ядром для работы с не моторизованными USB ридеами.

Возвращаемые значения:
1 – операция завершилась успешно.
0 – операция завершилась не удачей.

```
WORD WINAPI SendApduOut(BYTE * Cmd, WORD * Stat, BYTE *
 answer,
 WORD answer_max_len);
```

Описание:
Посылает APDU команду и получает ответ.

Параметры:
BYTE * Cmd – буфер, содержащий APDU команду.
WORD * Stat – возвращается статус, полученный от карты на APDU команду.
BYTE * answer – буфер для ответа от карты на APDU команду.

Возвращаемые значения:
Возвращается длинна полученного ответа от карты.

```
WORD WINAPI SendApduIn(BYTE * Cmd, BYTE * Data, WORD *
 Data_len,
 BYTE * answer, WORD answer_max_len);
```

Описание:
Посылает APDU команду и получает ответ.

Параметры:
BYTE * Cmd – буфер, содержащий APDU команду.
BYTE * Data – буфер, содержащий данные.
WORD * Data_len – длина данных. Возвращается длина полученных данных от карты.
BYTE * answer – буфер для ответа от карты на APDU команду.

Возвращаемые значения:
Возвращается статус, полученный от карты на APDU команду.

Управление считывателем smart-card.

В EMV библиотеке предусмотренно, два варианта управления считывателем:
1) Посредством динамичеси присоединяемой библиотеки. (SmartCard.dll)
2) Посредством функций обратного вызова.

1) Управление считывателем посредством динамически присоединяемой библиотеки:

В теле функции emv_kernelInit(), библиотека EMV пытается загрузить библиотеку SmartCard.dll.
Если библиотека отсутствует, или отсутствует минимальный набор функций,

возвращается указатель на FALSE и работа EMV библиотеки не может быть продолжена.

Библиотека SmartCard.dll должна содержать следующий минимальный набор функций:

BYTE WINAPI SelectReader(LPSTR rname,DWORD rname_size);

BYTE WINAPI OpenReader(LPSTR rname);

BYTE WINAPI CloseReader(void);

```
/**
*****************************************************************************************
* @fn BYTE WINAPI Atr(BYTE *atr, DWORD * max_len, BYTE * protocol)
* @b Назначение: \n
* @brief Получает ATR, используется EMV ядром. \n
* @author Ilya Yalin
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Посылает ридеру команду подать питание на карточку.
* 2.) Получает от карточки ATR.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b DWORD * max_len; - Максимальный размер буфера для приёма данных ATR.
*
* @param Выход: \n
* @b BYTE *atr - Указатель на буфер для данных ATR.
* @b DWORD * max_len; - Размер полученных данных.
* @b BYTE * protocol; - Адрес для получения протокола. Для моторизованных ридеров всегда 0.
*****************************************************************************************
* @return
* @b BYTE; - Статус выполнения операций.
* @b 1 – операция завершилась успешно.
* @b 0 – операция завершилась не удачей.
*****************************************************************************************
*/
BYTE WINAPI Atr (BYTE *atr, DWORD * max_len,BYTE * protocol);
/**
*****************************************************************************************
* @fn WORD WINAPI SendApduIn(BYTE *Cmd,BYTE * Data,WORD *Data_len,BYTE *answer,WORD
answer_max_len)
* @b Назначение: \n
* @brief Посылает APDU команду с данными и получает ответ. Используется EMV ядром. \n
* @author
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Посылает APDU команду и получает ответ.
* 2.) Возвращает данные в буфер.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b BYTE * Cmd; – Буфер, содержащий APDU команду.
* @b BYTE * Data; – Буфер, содержащий данные.
* @b WORD *Data_len; - Длина даннных.
* @b WORD answer_max_len; - Размер буфера для ответа.
*
* @param Выход: \n
* @b BYTE * answer; – Буфер для ответа от карты на APDU команду.
* @b WORD *Data_len; - Длина полученых от карты даннных.
*****************************************************************************************
* @return
* @b WORD; - Статус, полученный от карты на APDU команду.
*****************************************************************************************
*/
WORD WINAPI SendApduIn(BYTE *cmd,BYTE * data,WORD *data_len,BYTE *answer,WORD answer_max_len);
```

```
/**
*****************************************************************************************
* @fn WORD WINAPI SendApduOut(BYTE *Cmd,WORD *Stat,BYTE *answer,WORD answer_max_len)
* @b Назначение: \n
* @brief Посылает APDU команду и получает ответ. Используется EMV ядром. \n
* @author
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Посылает APDU команду и получает ответ.
* 2.) Возвращает данные в буфер.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b BYTE * Cmd; – Буфер, содержащий APDU команду.
* @b WORD answer_max_len; - Размер буфера для ответа.
*
* @param Выход: \n
* @b WORD * Stat; – Возвращается статус, полученный от карты на APDU команду.
* @b BYTE * answer; – Буфер для ответа от карты на APDU команду.
*****************************************************************************************
* @return
* @b WORD; - Длинна полученного ответа от карты.
*****************************************************************************************
*/
WORD WINAPI SendApduOut(BYTE * cmd, WORD * stat, BYTE * answer, WORD answer_max_len);
/**
*****************************************************************************************
* @fn BYTE WINAPI CheckConnect()
* @b Назначение: \n
* @brief Проверяет наличие карты в ридере. \n
* @author
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Проверяет флаг наличия карты в ридере. Флаг выставляется после выполнения TakeCard(), если
была подача карты.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b None
*
* @param Выход: \n
* @b None
*****************************************************************************************
* @return
* @b BYTE; - Результат выполнения.
* @b 1 - Карта присутствует.
* @b 0 - Карта отсуттствует.
*****************************************************************************************
*/
BYTE WINAPI CheckConnect();
```

Фнкции:

BYTE WINAPI SelectReader(LPSTR rname,DWORD rname_size);

BYTE WINAPI OpenReader(LPSTR rname);

BYTE WINAPI CloseReader(void);

В настоящее время не используются, и оставлены для возможного использования в будущем.

2) Управление считывателем через функции обратного вызова:
Макроопределение **USE_CALLBACK_FOR_CARD_READER** позволяет собрать демонстрационный проект, в котором управление считывателем осуществляется через функции обратного вызова.
Макроопределение **USE_CALLBACK_FOR_CARD_READER** должно быть установленно одновременно, в начале файлов Forms1.cs and EmvKerDll.cs.
Макроопределение **USE_DEMO_SMART_CARD_SIMULATOR_DLL** должно быть "выключено".
Для управления считывателем, библиотека EMV вызывает седующие функции обратного вызова:
Для управления считывателем, библиотека EMV вызывает седующие функции обратного вызова:

```
/**
*****************************************************************************************
* @fn WORD WINAPI emv_ApduIn(BYTE *Cmd,BYTE * Data,WORD *Data_len,BYTE *answer,WORD
answer_max_len)
* @b Назначение: \n
* @brief Посылает APDU команду с данными и получает ответ. Используется EMV ядром. \n
* @author
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Посылает APDU команду и получает ответ.
* 2.) Возвращает данные в буфер.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b BYTE * Cmd; – Буфер, содержащий APDU команду.
* @b BYTE * Data; – Буфер, содержащий данные.
* @b WORD *Data_len; - Длина даннных.
* @b WORD answer_max_len; - Размер буфера для ответа.
*
* @param Выход: \n
* @b BYTE * answer; – Буфер для ответа от карты на APDU команду.
* @b WORD *Data_len; - Длина полученых от карты даннных.
*****************************************************************************************
* @return
* @b WORD; - Статус, полученный от карты на APDU команду.
*****************************************************************************************
*/
```

public static unsafe ushort emv_ApduIn(byte* cmd, byte* data, ushort* data_len, byte* answer, ushort answer_max_len);

```
/**
*****************************************************************************************
* @fn WORD emv_ApduOut(BYTE *Cmd,WORD *Stat,BYTE *answer,WORD answer_max_len)
* @b Назначение: \n
* @brief Посылает APDU команду и получает ответ. Используется EMV ядром. \n
* @author
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Посылает APDU команду и получает ответ.
* 2.) Возвращает данные в буфер.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b BYTE * Cmd; – Буфер, содержащий APDU команду.
* @b WORD answer_max_len; - Размер буфера для ответа.
*
* @param Выход: \n
* @b WORD * Stat; – Возвращается статус, полученный от карты на APDU команду.
* @b BYTE * answer; – Буфер для ответа от карты на APDU команду.
*****************************************************************************************
* @return
* @b WORD; - Длинна полученного ответа от карты.
*****************************************************************************************
*/
```

public static unsafe ushort emv_ApduOut(byte* cmd, ushort* stat, byte* answer, ushort answer_max_len);

```
/**
*****************************************************************************************
* @fn BYTE emv_Atr(BYTE *atr, DWORD * max_len, BYTE * protocol)
* @b Назначение: \n
* @brief Получает ATR, используется EMV ядром. \n
* @author Ilya Yalin
*****************************************************************************************
* @b Выполняет:
* @code
```

```
* 1.) Посылает ридеру команду подать питание на карточку.
* 2.) Получает от карточки ATR.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b DWORD * max_len; - Максимальный размер буфера для приёма данных ATR.
*
* @param Выход: \n
* @b BYTE *atr - Указатель на буфер для данных ATR.
* @b DWORD * max_len; - Размер полученных данных.
* @b BYTE * protocol; - Адрес для получения протокола. Для моторизованных ридеров всегда 0.
*****************************************************************************************
* @return
* @b BYTE; - Статус выполнения операций.
* @b 1 – операция завершилась успешно.
* @b 0 – операция завершилась не удачей.
*****************************************************************************************
*/
```

public static unsafe byte emv_Atr(byte* dest_atr, uint* max_len, byte* protocol);

```
/**
*****************************************************************************************
* @fn BYTE emv_CheckConnect()
* @b Назначение: \n
* @brief Проверяет наличие карты в ридере. \n
* @author
*****************************************************************************************
* @b Выполняет:
* @code
* 1.) Проверяет флаг наличия карты в ридере. Флаг выставляется после выполнения TakeCard(),
* если была подача карты.
* @endcode
*****************************************************************************************
* @param Вход: \n
* @b None
*
* @param Выход: \n
* @b None
*****************************************************************************************
* @return
* @b BYTE; - Результат выполнения.
* @b 1 - Карта присутствует.
* @b 0 - Карта отсуттствует.
*****************************************************************************************
*/
```

private static byte emv_CheckConnect();

Перед первым использованием, функции обратного вызова,должны быть зарегистрированны в EMV библиотеке:

```
emv_kernel_registry_ApduIn (hEmvInit, _emv_ApduIn);
emv_kernel_registry_ApduOut (hEmvInit, _emv_ApduOut);
emv_kernel_registry_Atr (hEmvInit, _emv_Atr);
emv_kernel_registry_CheckConnect (hEmvInit, _emv_CheckConnect);
```