Advanced Card Systems Ltd.

ACR80 Smart Card Terminal

EXTENDED PC/SC APIs

Version 1.01 01-2006

Unit 1008, 10th Floor, Hongkong International Trade and Exhibition Centre
1 Trademart Drive, Kowloon Bay, Hong Kong
Tel: +852 2796 7873 Fax: +852 2796 1286 Email: info@acs.com.hk Website: www.acs.com.hk

Contents

1. List of APIs

The following APIs are supported by ACR80 PC/SC driver:

- IOCTL_SMARTCARD_DIRECT (2050)
- IOCTL_SMARTCARD_SELECT_SLOT (2051)
- IOCTL_SMARTCARD_DRAW_LCDBMP (2052)
- IOCTL_SMARTCARD_DISPLAY_LCD (2053)
- IOCTL_SMARTCARD_CLR_LCD (2054)
- IOCTL_SMARTCARD_READ_KEYPAD (2055)
- IOCTL_SMARTCARD_READ_MAGSTRIP (2056)
- IOCTL_SMARTCARD_READ_RTC (2057)
- IOCTL_SMARTCARD_SET_RTC (2058)
- IOCTL_SMARTCARD_SET_OPTION (2059)
- IOCTL_SMARTCARD_SET_LED (2060)
- IOCTL_SMARTCARD_USE_ENCRYPTION (2061)
- IOCTL_SMARTCARD_LOAD_KEY (2062)
- IOCTL_SMARTCARD_COMPUTE_MAC (2063)
- IOCTL_SMARTCARD_DECRYPT_MAC (2064)
- IOCTL_SMARTCARD_READ_EEPROM (2065)
- IOCTL_SMARTCARD_WRITE_EEPROM (2066)
- IOCTL_SMARTCARD_GET_VERSION (2067)
- IOCTL_SMARTCARD_DUKPT_INIT_KEY (2068)
- IOCTL_SMARTCARD_REQ_DUKPT_PIN (2069)
- IOCTL_SMARTCARD_ABORT_DUKPT_PIN (2070)
- IOCTL_SMARTCARD_SET_USB_VIDPID (2071)

2. Descriptions of APIs

2.1 [IOCTL_SMARTCARD_DIRECT]

This API allows the PC/SC application to send/receive data to/from ACR80 directly. The application is responsible for building the ACR80 command according to the reader's protocol.

INPUT:

- Input buffer size must not be larger than 600 bytes

OUTPUT:

- Output buffer stores the returned data and reader status. The size must be at least 2 bytes

EXAMPLE:

```
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_DIRECT,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

Note: PC/SC application should connect to the smart card using exclusive direct access.

2.2 [IOCTL_SMARTCARD_SELECT_SLOT]

This API allows the PC/SC application to select the card slot to be used in the subsequence communication.

INPUT:

- Input buffer size must be 1 byte and stores the new slot to use

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_SELECT_SLOT,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.3 [IOCTL_SMARTCARD_DISPLAY_LCD]

This API will instruct the driver to send the bitmap pattern stored in its internal buffer or a predefined character map to the reader's LCD panel.

INPUT:

- If input buffer size is 0, then driver internal bitmap pattern will be sent to LCD panel
- If input buffer size is 2, then the 1st byte will determine to offset to display the character and the 2nd byte is an index to which character will be displayed.

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_DISPLAY_LCD,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.4 [IOCTL_SMARTCARD_DRAW_LCDBMP]

This API instructs the driver to convert the LCD Map buffer in AC_GRAPHICS structure into a format that can be displayed on ACR80 LCD display.

```
typedef struct {
      BYTE  StartXPos;        // starting position on x-coordinate
      BYTE  StartYPos;        // starting position on y-coordinate
      BYTE  Type;             // type of graphic
      BYTE  Map[488];         // ACSII string or bit-map of graphic
} AC_GRAPHICS, *PAC_GRAPHICS;
```

INPUT:

- AC_GRAPHICS.StartXPos specifies the starting X-coordinate of LCD bitmap or alphanumeric text.
- AC_GRAPHICS.StartYPos specifies the starting Y-coordinate of LCD bitmap or alphanumeric text.
- AC_GRAPHICS.Type specifies the type of data stored in AC_GRAPHICS.Type. Possible values are:
    - AC_ALPHANUMERIC (0x00) – to display numbers and text using pre-defined fonts
    - AC_MAP (0x01) – to display bitmap pattern of arbitrary size.
    - AC_FULL_MAP (0x02) – to display a full-LCD-sized bitmap pattern. With this option, AC_GRAPHICS.StartXPos and AC_GRAPHICS.StartYPos will be neglected.

- AC_GRAPHICS.Map stores the alphanumeric text or bitmap pattern to be displayed on ACR80 LCD panel.

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_DRAW_LCDBMP,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.5 [IOCTL_SMARTCARD_CLR_LCD]

This API allows the PC/SC application to request the driver to clear the content of specific LCD pages.

INPUT:

- Input buffer is 1 byte in size and its bitmask indicates which pages to be erased.

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
// to erase page 1 and page 3 of LCD display
lpInBuffer[0] = 0x05;   // 00000101b
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_CLR_LCD,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.6 [IOCTL_SMARTCARD_READ_KEYPAD]

This API allows the PC/SC application to read key input from reader's keypad.

INPUT:

- Input buffer specifies the timeout in sec of the key-press event.
- Input buffer size should be at least 1 byte.

OUTPUT:

- Output buffer stores the key value and returned reader status. The size must be at least 3 bytes

EXAMPLE:

```
lpInBuffer[0] = 0x10;    // 16 sec timeout
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_READ_KEYPAD,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.7 [IOCTL_SMARTCARD_READ_MAGSTRIP]

This API allows the PC/SC application to read data from magnetic stripe on the reader.

INPUT:

- Byte 0 of input buffer specifies the bitmask track number to read.
- Byte 1 of input buffer specifies the timeout used in reading the magnetic card.
- Input buffer size should be at least 2 bytes.

OUTPUT:

- Output buffer stores the output data read from the magnetic card and returned reader status. The size must be at least 2 + sizeof(AC_MAGNETIC_CARD) bytes

EXAMPLE:

```
AC_MAGNETIC_CARD*  pMagCard;
lpInBuffer[0] = 0x03;    // 00000011b track 1 and 2
lpInBuffer[1] = 0x10;    // 16 sec timeout
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_READ_MAGSTRIP,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);

pMagCard = (AC_MAGNETIC_CARD*) lpOutBuffer
…
```

2.8 [IOCTL_SMARTCARD_READ_RTC]

This API allows the PC/SC application to read time and date from the real-time clock in the reader.

INPUT:

- None

OUTPUT:

- Output buffer stores the output data read from the real-time clock and returned reader status. The size must be at least 2 + sizeof(AC_READER_CLOCK) bytes

EXAMPLE:

```
AC_READER_CLOCK *  pRTC;
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_READ_MAGSTRIP,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);

pRTC = (AC_READER_CLOCK*) lpOutBuffer;
```

2.9 [IOCTL_SMARTCARD_SET_RTC]

This API allows the PC/SC application to set time and date to the real-time clock in the reader.

INPUT:

- Input buffer is the structure, AC_READER_CLOCK
- Input buffer length is at least sizeof(AC_READER_CLOCK)

OUTPUT:

- Output buffer stores returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
AC_READER_CLOCK *  pRTC;
pRTC = (AC_READER_CLOCK*) lpInBuffer;
pRTC->Year = 02;
pRTC->Month = 11;
pRTC->Date = 08;         // 8th Nov
pRTC->Day = 5;           // Friday
pRTC->Hour = 14;
pRTC->Minute = 19;
pRTC->Second = 0;
```

```
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_SET_RTC,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.10 [IOCTL_SMARTCARD_SET_OPTION]

This API allows the PC/SC application to set certain reader operation options.

INPUT:

- Input buffer is a 1-byte option bitmask.

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
lpInBuffer[0] = 0x03;   // 00000011b – turn on buzzer and LCD backlight
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_SET_OPTION,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.11 [IOCTL_SMARTCARD_SET_LED]

This API allows the PC/SC application to control the card slot status LED lights.

INPUT:

- Input buffer is a 1-byte option bitmask.

  (Note*: ONLY bit0-bit3 is used. Please refer to ACR50 Reference manual for detail)

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes

EXAMPLE:

```
lpInBuffer[0] = 0x09;   // 00001001b - turn on slot 2 (red light) and slot 1
(green light) LED
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_SET_LED,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.12 [IOCTL_SMARTCARD_USE_ENCRYPTION]

This API allows the PC/SC application to instruct reader to perform cryptographic function on the input data with pre-defined encryption keys.

INPUT:

- Input buffer is in the format of FUNC+INPUT. FUNC is 1-byte function byte and INPUT is an 8-byte data to be encrypted/decrypted.

  FUNC:

  0x08: 3DES encryption

  0x88: 3DES decryption

- The input buffer size must be at least 9 bytes.

OUTPUT:

- Output buffer stores the encrypted/decrypted message and returned reader status. The size must be at least 2 bytes + length of the cipher/plain message.

EXAMPLE:

```
lpInBuffer[0] = 0x08;   // Encrypt the input data using 3DES
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_USE_ENCRYPTION,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.13 [IOCTL_SMARTCARD_LOAD_KEY]

This API allows the PC/SC application to load any encryption keys used in the cryptographic functions

INPUT:

- Input buffer is in a format of TYPE+KEY. TYPE is a 1-byte value classifying the type of key to be loaded. KEY is a 16-byte key to be loaded to the reader.

  TYPE: 0x00 – KEK; 0x01– Kc; 0x02 – MAC

  Please refer to ACR50 Reference manual for further detail.

- The input buffer length must be at least 17 bytes

OUTPUT:

- Output buffer stores the returned reader status. The size must be at least 2 bytes.

EXAMPLE:

```
lpInBuffer[0] = 0x08;   // Encrypt the input data using 3DES
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_LOAD_KEY,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.14 [IOCTL_SMARTCARD_COMPUTE_MAC]

This API allows the PC/SC application to instruct the reader to compute MAC on message with MAC key using 3-DES CBC.

INPUT:

- Input buffer is in a format of LEN+MESSAGE. LEN is a 1-byte value specifying the number of bytes of the following message. The message length can range from 1-256 (with LEN = 0 to represent 256). MESSAGE is the message to be computed on.

- The input buffer size is 1+LEN bytes.

OUTPUT:

- Output buffer is in the format of LEN+MAC and 2-byte status word from reader. LEN is 1-byte value specifying the length of the encrypted MAC message. MAC is the computed MAC messages. The output buffer size is 3 + LEN bytes.

EXAMPLE:

```
lpInBuffer[0] = 0x0A;   // Length of the MAC message
CopyMemory(&lpInBuffer[1],"EncyrptMAC");
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_COMPUTE_MAC,
```

```
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.15 [IOCTL_SMARTCARD_DECRYPT_MAC]

This API allows the PC/SC application to instruct the reader to decrypt MAC-encrypted message with MAC key using 3-DES CBC.

INPUT:

- Input buffer is in a format of LEN+MESSAGE. LEN is a 1-byte value specifying the number of bytes of the following message. The message length can range from 1-256 (with LEN = 0 to represent 256). MESSAGE is the MAC-encrypted message to be decrypted.
- The input buffer size is 1+LEN bytes.

OUTPUT:

- Output buffer is in the format of LEN+MAC. LEN is a 1-byte value specifying the length of the decrypted message. MAC stores the MAC-decrypted message and returned reader status.
- The output buffer size is 3 + LEN bytes.

EXAMPLE:

```
lpInBuffer[0] = 0x0A;   // Length of the MAC message
CopyMemory(&lpInBuffer[1],"EncyrptMAC");
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_DECRYPT_MAC,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.16 [IOCTL_SMARTCARD_READ_EEPROM]

This API allows the PC/SC application to read the content of built-in EEPROM on the reader.

INPUT:

- Input buffer is in a format of LEN+ADR(HI)+ADR(LOW). LEN is a 1-byte value specifying the number of bytes to read. The data length can range from 1-256 (with LEN = 0 to represent 256). ADR(HI)+ADR(LOW) are 1-byte length each and specify the read address (0x0100 – 0x7FFF).
- The input buffer size is 3 bytes.

OUTPUT:

- Output buffer is in the format of LEN+DATA and a 2-byte reader status. LEN is a 1-byte value and specifies the length of data returned. DATA is the EEPROM content in specified length.
- Output buffer size is 3+LEN bytes

EXAMPLE:

```
lpInBuffer[0] = 0x0A;   // read 10 bytes
lpInBuffer[1] = 0x01;
lpInBuffer[2] = 0x50;   // read from addr 0x0150
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_READ_EEPROM,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.17 [IOCTL_SMARTCARD_WRITE_EEPROM]

This API allows the PC/SC application to write data to the built-in EEPROM on the reader.

INPUT:

- Input buffer is in a format of LEN+ADR(HI)+ADR(LOW)+DATA. LEN is a 1-byte value specifying the number of bytes to read. The data length can range from 1-256 (with LEN = 0 to represent 256). ADR(HI)+ADR(LOW) are 1-byte length each and specify the write address (0x0100 – 0x7FFF).
- The input buffer size is 3+LEN bytes.

OUTPUT:

- Output buffer is a 2-byte status word from reader.
- Output buffer size is at least 2 bytes.

EXAMPLE:

```
lpInBuffer[0] = 0x0A;   // write 5 bytes
lpInBuffer[1] = 0x01;
lpInBuffer[2] = 0x50;   // read from addr 0x0150
lpInBuffer[3] = 0x05;
lpInBuffer[4] = 0x05;
lpInBuffer[5] = 0x05;
lpInBuffer[6] = 0x05;
lpInBuffer[7] = 0x05;
LONG SCardControl(
  hCard,
```

```
  IOCTL_SMARTCARD_WRITE_EEPROM,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.18 [IOCTL_SMARTCARD_GET_VERSION]

This API allows the PC/SC application to query the version of driver and reader firmware.

INPUT:

- None

OUTPUT:

- Output buffer is a structure, AC_VERSION_INFO and a 2-byte status word from reader.
- Output buffer size is at least 2 + sizeof(AC_VERSION_INFO).

EXAMPLE:

```
AC_VERSION_INFO*  pVer;
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_GET_VERSION,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);

pVer = (AC_VERSION_INFO*) lpOutBuffer;
cout << pVer->OptionRegister;
cout << pVer->FwVersion;
cout << pVer->Max_C;
cout << pVer->Max_R;
cout << pVer->DrVersion;
```

2.19 [IOCTL_SMARTCARD_DUKPT_INIT_KEY]

This API loads the Initial Pin Encryption Key (IPEK) and Initial Key Serial Number (IKSN) to the reader.

INPUT:

- Input buffer is a 18-byte data buffer
- The format consists of IPEK (first 8 bytes) + IKSN (last 10 bytes)

OUTPUT:

- None

EXAMPLE:

```
CopyMemory(lpBuffer,"\x6A\xC2\x92\xFA\xA1\x31\x5B\x4D",8);  // IPEK
CopyMemory(&lpBuffer[8],"\xFF\xFF\x98\x76\54\x32\x10\xE0\x00\x00",10);  //ISKN
lpInBufferSzie = 18;
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_DUKPT_INIT_KEY,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.20 [IOCTL_SMARTCARD_REQ_DUKPT_PIN]

This API instructs the reader to get the DUKPT PIN from keypad and returns the encrypted pin to the application

INPUT:

- Input buffer is a 6-byte account number.
- Input buffer size must be at least 6.

OUTPUT:

- Output buffer is in the format of, LEN+ENC_PIN. LEN specifies the number of bytes in the following ENC_PIN block. ENC_PIN is the encrypted pin.
- Output buffer size is at least 3+LEN.

EXAMPLE:

```
CopyMemory(lpBuffer,"012345",6);    // Account Number
lpInBufferSzie = 6;
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_REQ_DUKPT_PIN,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.21 [IOCTL_SMARTCARD_ABORT_DUKPT_PIN]

This API cancels the previous REQ_DUKPT_PIN request and disables the keypad

INPUT:

- None

OUTPUT:

- Output buffer stores the reader status word.
- Output buffer size is at least 2.

EXAMPLE:

```
CopyMemory(lpBuffer,"012345",6);     // Account Number
lpInBufferSzie = 6;
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_ABORT_DUKPT_PIN,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```

2.22 [IOCTL_SMARTCARD_SET_USB_VIDPID]

This API allows the PC/SC application to set the idVendor and idProduct values of the USB standard Device Descriptor of the reader to any desired values (except FFFFh). The new setting will take effect starting from the next reader power up.

INPUT:

- Input buffer is in the format of, VID_LOW+VID_HIGH+PID_LOW+PID_HIGH
  VID_LOW: Low byte of VID
  VID_HIGH: High byte of VID
  PID_LOW: Low byte of PID
  PID_HIGH: High byte of PID
- Input Buffer size should

OUTPUT:

- Output buffer stores the reader status word.
- Output buffer size is at least 2.

EXAMPLE:

```
lpInBuffer[0] = 0x01;
lpInBuffer[1] = 0x2F;    // VID: 2F01h
lpInBuffer[2] = 0x05;
lpInBuffer[3] = 0x60;    // PID: 6005h
lpInBufferSize = 4;
LONG SCardControl(
  hCard,
  IOCTL_SMARTCARD_SET_USB_VIDPID,
  lpInBuffer,
  nInBufferSize,
  lpOutBuffer,
  nOutBufferSize,
  lpBytesReturned
);
```