Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1281S-C1 Serial Dual Interface Reader

Application Programming Interface V1.01

Subject to change without prior notice

info@acs.com.hk
www.acs.com.hk

Table of Contents

List of Tables

1.0. Introduction

This manual describes the use of ACR1281S-C1 interface software to facilitate application development with the ACR1281S-C1 reader. This interface software is supplied in the form of 32-bit and 64-bit DLL (Dynamic Link Library), which can be programmed using popular development tools like Java, Delphi, Visual Basic, Visual C++, Visual C# and Visual Basic .Net.

The ACR1281S-C1 DLL is a set of high-level functions provided for the application software use. It supplies a consistent API (Application Programming Interface) for the application to operate on ACR1281S-C1 and on the corresponding presented card. The DLL communicates with the ACR1281S-C1 via the communication port facilities provided by the operating system.

The ACR1281S-C1 API defines a common way of accessing the ACR1281S-C1. Application programs invoke the ACR1281S-C1 through the interface functions and perform operations on the presented card.

The header file "acr.h" is available for the program developer, which contains all the function prototypes and macros as described in this document.

1.1. Features

- Serial RS232 Interface: Baud Rate = 9.6 kbps (default), 19.2 kbps, 38.4 kbps, 57.6 kbps, 115.2 kbps, 230.4 kbps
- USB interface for power supply
- CCID-like frame format (binary format)
- Contactless Smart Card Reader:
  - Read/write speed of up to 848 kbps
  - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
  - Supports ISO 14443 Part 4 Type A and B cards and Mifare series
  - Built-in anti-collision feature (only one tag is accessed at any time)
  - Supports extended APDU (max. 64 kbytes)
- Contact Smart Card Reader:
  - Supports ISO 7816 Class A, B and C (5 V, 3V and 1.8 V)
  - Supports microprocessor cards with T=0 or T=1 protocol
  - Supports memory cards
  - ISO 7816 compliant SAM slot
- Built-in Peripherals:
  - Two user-controllable LEDs
  - User-controllable buzzer
- USB Firmware Upgradability
- Compliant with the following standards:
  - ISO 14443
  - ISO 7816
  - CE
  - FCC
  - RoHS

2.0. ACR API

2.1. Using ACR API

To use ACR API, the source code must include a header file "acr.h".

Source Code Example:

```
#include <stdio.h>
#include <acr.h>

int main(int argc, char *argv[])
{
   HANDLE hReader;
   DWORD ret;

   // Open reader using COM1
   ret = ACR_Open(TEXT("\\\\.\\COM1"), &hReader);
   if (ret != ERROR_SUCCESS)
   {
      printf("ACR_Open failed with error 0x%08X\n", ret);
      return 1;
   }

   // TODO: Place other API function calls here
   // ...

   // Close reader
   ret = ACR_Close(hReader);
   if (ret != ERROR_SUCCESS)
      printf("ACR_Close failed with error 0x%08X\n", ret);

   return 0;
 }
```

2.1.1. Opening Reader

On Windows, port name is specified as "\\.\COMx" where x is the port number. e.g., "\\.\COM1" is the first serial port.

On Linux, port name can be any file under directory "/dev". Standard Linux serial port driver creates the device file as "/dev/ttySx" where x is the port number. e.g., "/dev/ttyS0" is the first serial port.

`ACR_Open()` will return a handle of reader in parameter if the port is opened successfully.

2.1.2. Closing Reader

After finishing the operation, use `ACR_Close()` to close the reader.

2.1.3. Using ANSI and Unicode Functions

If your source code defined UNICODE macro, the following functions will be mapped to Unicode version.

`ACR_Open()` -> `ACR_OpenW()`

Otherwise, it will be mapped to ANSI version.

`ACR_Open()` -> `ACR_OpenA()`

2.2. Reader

2.2.1. Define Documentation

2.2.1.1. ACR_Open and ACR_OpenA

Format:

```
#define ACR_Open ACR_OpenA
```

`ACR_Open` will be mapped to `ACR_OpenW()` function if UNICODE is defined.

Otherwise, it will be mapped to `ACR_OpenA()` function.

2.2.2. Function Documentation

2.2.2.1. ACR_OpenA

Format:

```
DWORD WINAPI ACR_OpenA        ( LPCSTR          portName,
                                 LPHANDLE        phReader
                               )
```

Function Description: Open reader (ANSI).

This function opens a reader and returns a handle value as reference.

Parameters	Description	
`[in] portName`	Port name. For Windows Platform, the port name is specified as "\\.\COMx" where x is the port number. E.g., "\\.\COM1" is the first serial port. For Linux Platform, the port name can be any file under directory "/dev". Standard Linux serial port driver creates the device file as "/dev/ttySx" where x is the port number. E.g., "/dev/ttyS0" is the first serial port.	
`[out] phReader`	Pointer to the `HANDLE` variable.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file "errno.h" and ACR error codes.

Table 1: ACR_OpenA Function Description

Source Code Example:

```
HANDLE hReader;
DWORD ret;

// Open reader using COM1
ret = ACR_Open(TEXT("\\\\.\\COM1"), &hReader);
if (ret != ERROR_SUCCESS)
   printf("ACR_Open failed with error 0x%08X\n", ret);
```

2.2.2.2. ACR_OpenW

Format:

```
DWORD WINAPI ACR_OpenW        ( LPCWSTR          portName,
                                 LPHANDLE         phReader
                               )
```

Function Description: Open reader (Unicode).

This function opens a reader and returns a handle value as reference.

Parameters	Description	
`[in] portName`	Port name. For Windows Platform, the port name is specified as "\\.\COMx" where x is the port number. E.g., "\\.\COM1" is the first serial port. For Linux Platform, the port name can be any file under directory "/dev". Standard Linux serial port driver creates the device file as "/dev/ttySx" where x is the port number. E.g., "/dev/ttyS0" is the first serial port.	
`[out] phReader`	Pointer to the `HANDLE` variable.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file "errno.h" and ACR error codes.

Table 2: ACR_OpenW Function Description

Source Code Example:

```
HANDLE hReader;
DWORD ret;

// Open reader using COM1
ret = ACR_Open(TEXT("\\\\.\\COM1"), &hReader);
if (ret != ERROR_SUCCESS)
   printf("ACR_Open failed with error 0x%08X\n", ret);
```

2.2.2.3. ACR_Close

Format:

```
DWORD WINAPI ACR_Close         ( HANDLE     hReader )
```

Function Description: Close reader.

This function closes the reader and releases the resources.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR_Open()` function.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 3: ACR_Close Function Description

Source Code Example:

```
DWORD ret;

// Close reader
ret = ACR_Close(hReader);
if (ret != ERROR_SUCCESS)
   printf("ACR_Close failed with error 0x%08X\n", ret);
```

2.2.2.4. ACR_GetNumSlots

Format:

```
DWORD WINAPI ACR_GetNumSlots  ( HANDLE     hReader,
                                 LPDWORD    pNumSlots
                               )
```

Function Description: Get number of slots.

This function retrieves the number of slots of the reader.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR_Open()` function.	
`[out] pNumSlots`	Pointer to a `DWORD` variable in which the number of slots is returned.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 4: ACR_GetNumSlots Function Description

Source Code Example:

```
DWORD numSlots;
DWORD ret;

// Get number of slots
ret = ACR_GetNumSlots (hReader, &numSlots);
if (ret != ERROR_SUCCESS)
   printf("ACR_GetNumSlots failed with error 0x%08X\n", ret);
```

2.2.2.5. ACR_GetBaudRate

Format:

```
DWORD WINAPI ACR_GetBaudRate ( HANDLE        hReader,
                                LPDWORD       pBaudRate
                              )
```

Function Description: Get baud rate.

This function retrieves the baud rate of the reader.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR_Open()` function.	
`[out] pBaudRate`	Pointer to a `DWORD` variable in which the baud rate is returned	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.

Parameters		Description
	Failure	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 5: ACR_GetBaudRate Function Description

Source Code Example:

```
DWORD baudRate;
DWORD ret;

// Get baud rate
ret = ACR_GetBaudRate(hReader, &baudRate);
if (ret != ERROR_SUCCESS)
   printf("ACR_GetBaudRate failed with error 0x%08X\n", ret);
```

2.2.2.6. ACR_SetBaudRate

Format:

```
DWORD WINAPI ACR_SetBaudRate ( HANDLE          hReader,
                               DWORD           baudRate
                             )
```

Function Description: Set baud rate.

This function sets the baud rate of the reader. ACR1281S-C1 reader supports 9600, 19200, 38400, 57600, 115200, 128000, 230400, 250000, and 500000 bps.

Parameters		Description
`[in] hReader`	A reference value returned from `ACR_Open()` function.	
`[in] baudRate`	Baud rate.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 6: ACR_SetBaudRate Function Description

Source Code Example:

```
DWORD ret;

// Set baud rate to 115200 bps
ret = ACR_SetBaudRate(hReader, 115200);
if (ret != ERROR_SUCCESS)
   printf("ACR_SetBaudRate failed with error 0x%08X\n", ret);
```

2.2.2.7. ACR_GetTimeouts

Format:

```
DWORD WINAPI ACR_GetTimeouts ( HANDLE            hReader,
                               PACR_TIMEOUTS     pTimeouts
                             )
```

Function Description: Get timeouts.

This function retrieves the timeout parameters for status and response operations of the reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR_Open() function.	
[out] pTimeouts	Pointer to an ACR_TIMEOUTS structure in which the timeout information is returned.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 7: ACR_GetTimeouts Function Description

Source Code Example:

```
ACR_TIMEOUTS timeouts;
DWORD ret;

// Get timeouts
ret = ACR_GetTimeouts (hReader, &timeouts);
if (ret != ERROR_SUCCESS)
   printf("ACR_GetTimeouts failed with error 0x%08X\n", ret);
```

2.2.2.8. ACR_SetTimeouts

Format:

```
DWORD WINAPI ACR_SetTimeouts ( HANDLE                 hReader,
                                const PACR_TIMEOUTS   pTimeouts
                              )
```

Function Description: Set timeouts.

This function sets the timeout parameters for status and response operations on the reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR_Open() function.	
[in] pTimeouts	Pointer to an ACR_TIMEOUTS structure that contains the new timeout values.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 8: ACR_SetTimeouts Function Description

Source Code Example:

```
ACR_TIMEOUTS timeouts;
DWORD ret;

// Get timeouts
// ...

// Modify status timeout to 100 ms
timeouts.statusTimeout = 100;

// Set timeouts
ret = ACR_SetTimeouts(hReader, &timeouts);
if (ret != ERROR_SUCCESS)
   printf("ACR_SetTimeouts failed with error 0x%08X\n", ret);
```

2.3. Card

2.3.1. Function Documentation

2.3.1.1. ACR_ExchangeApdu

Format:

```
DWORD WINAPI ACR_ExchangeApdu ( HANDLE          hReader,
                                DWORD           slotNum,
                                const LPBYTE    sendBuffer,
                                DWORD           sendBufferLen,
                                LPBYTE          recvBuffer,
                                LPDWORD         pRecvBufferLen
                              )
```

Function Description: Exchange APDU.

This function sends APDU command and receives APDU response from the card.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR_Open()` function.	
`[in] slotNum`	Slot number.	
`[in] sendBuffer`	A pointer to the actual data to be written to the card.	
`[in] sendBufferLen`	The length in number of bytes of the `sendBuffer` parameter.	
`[out] recvBuffer`	A pointer to any data returned from the card.	
`[in,out] pRecvBufferLen`	The length in number of bytes of the `recvBuffer` parameter and receives the actual number of bytes received from the card.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 9: ACR_ExchangeAPDU Function Description

Source Code Example:

```
BYTE command[] = { 0x80, 0x84, 0x00, 0x00, 0x08 };
DWORD commandLen = sizeof(command);
BYTE response[300];
DWORD responseLen = sizeof(response);
DWORD ret;

// Exchange APDU in slot 0
ret  =  ACR_ExchangeApdu(hReader,  0,  command,  commandLen,  response,
&responseLen);
if (ret != ERROR_SUCCESS)
   printf("ACR_ExchangeApdu failed with error 0x%08X\n", ret);
```

2.3.1.2. ACR_GetIccStatus

Format:

```
DWORD WINAPI ACR_GetIccStatus ( HANDLE          hReader,
                                 DWORD          slotNum,
                                 LPDWORD        pIccStatus
                               )
```

Function Description: Get ICC status.

This function returns the current ICC status in slot of the reader.

Parameters	Description
[in] hReader	A reference value returned from ACR_Open() function.
[in] slotNum	Slot number.
[out] pIccStatus	Pointer to a DWORD variable in which the ICC status is returned. **Value** / **Meaning** ACR_ICC_UNKNOWN: The current status is unknown. ACR_ICC_ABSENT: There is no card in the reader. ACR_ICC_PRESENT: There is a card in the reader. ACR_ICC_POWERED: Power is being provided to the card.

Return Value	ERROR_SUCCESS	The operation completed successfully.

Parameters		Description
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file "errno.h" and ACR error codes.

Table 10: ACR_GetIccStatus Function Description

Source Code Example:

```
DWORD iccStatus;
DWORD ret;

// Get ICC status in slot 0
ret = ACR_GetIccStatus(hReader, 0, &iccStatus);
if (ret != ERROR_SUCCESS)
   printf("ACR_GetIccStatus failed with error 0x%08X\n", ret);
```

2.3.1.3. ACR_PowerOffIcc

Format:

```
DWORD WINAPI ACR_PowerOffIcc  ( HANDLE          hReader,
                                 DWORD          slotNum
                               )
```

Function Description: Power off ICC in slot.

This function powers off the card in the slot.

Parameters		Description
`[in] hReader`		A reference value returned from `ACR_Open()` function.
`[in] slotNum`		Slot number.
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file "errno.h" and ACR error codes.

Table 11: ACR_PowerOffIcc Function Description

Source Code Example:

```
DWORD ret;

// Power off slot 0
ret = ACR_PowerOffIcc(hReader, 0);
if (ret != ERROR_SUCCESS)
   printf("ACR_PowerOffIcc failed with error 0x%08X\n", ret);
```

2.3.1.4. ACR_PowerOnIcc

Format:

```
DWORD WINAPI ACR_PowerOnIcc   ( HANDLE          hReader,
                                DWORD           slotNum,
                                LPBYTE          atr,
                                LPDWORD         pAtrLen
                              )
```

Function Description: Power on ICC in slot.

This function powers on the card in the slot and returns the ATR string from the card.

Parameters	Description	
`[in] hReader`	A reference value returned from `ACR_Open()` function.	
`[in] slotNum`	Slot number.	
`[out] atr`	A pointer to the buffer that receives the ATR string returned from the card.	
`[in,out] pAtrLen`	The length in number of bytes of the `atr` parameter and receives the actual number of bytes received from the card.	
Return Value	`ERROR_SUCCESS`	The operation completed successfully.
	`Failure`	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file “errno.h” and ACR error codes.

Table 12: ACR_PowerOnICC Function Description

Source Code Example:

```
BYTE atr[64];
DWORD atrLen = sizeof(atr);
DWORD ret;

// Power on slot 0
ret = ACR_PowerOnIcc(hReader, 0, atr, &atrLen);
if (ret != ERROR_SUCCESS)
   printf("ACR_PowerOnIcc failed with error 0x%08X\n", ret);
```

2.3.1.5. ACR_SetProtocol

Format:

```
DWORD WINAPI ACR_SetProtocol  ( HANDLE          hReader,
                                 DWORD           slotNum,
                                 DWORD           preferredProtocols,
                                 LPDWORD         pActiveProtocol
                              )
```

Function Description: Set protocol.

This function selects the protocol and parameters according to the ATR string returned from the card. Before setting the protocol, the card must be powered using `ACR_PowerOnIcc()` function.

Parameters	Description
`[in] hReader`	A reference value returned from `ACR_Open()` function.
`[in] slotNum`	Slot number.
`[in] preferredProtocols`	A bitmask of preferred protocols. Possible values may be combined with the OR operation. **Value** / **Meaning** ACR_PROTOCOL_T0 — T=0 is an acceptable protocol. ACR_PROTOCOL_T1 — T=1 is an acceptable protocol.

Parameters	Description	
[out] pActiveProtocol	A flag that indicates the established active protocol. **Value** / **Meaning** ACR_PROTOCOL_T0 — T=0 is the active protocol. ACR_PROTOCOL_T1 — T=1 is the active protocol. ACR_PROTOCOL_UNDEFINED — Protocol is undefined.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file "errno.h" and ACR error codes.

Table 13: ACR_SetProtocol Function Description

Source Code Example:

```
DWORD activeProtocol;
DWORD ret;

// Power on slot 0
// ...

// Set protocol to T=0 or T=1
ret = ACR_SetProtocol(hReader, 0, ACR_PROTOCOL_T0 | ACR_PROTOCOL_T1,
&activeProtocol);
if (ret != ERROR_SUCCESS)
   printf("ACR_SetProtocol failed with error 0x%08X\n", ret);
```

2.4. Peripheral

2.4.1. Function Documentation

2.4.1.1. ACR_Control

Format:

```
DWORD WINAPI ACR_Control     ( HANDLE         hReader,
                               DWORD          slotNum,
                               DWORD          controlCode,
                               LPCVOID        inBuffer,
                               DWORD          inBufferSize,
                               LPVOID         outBuffer,
                               DWORD          outBufferSize,
                               LPDWORD        pBytesReturned
                             )
```

Function Description: Control peripheral.

This function gives the user direct control on the peripherals in the reader.

Parameters	Description	
[in] hReader	A reference value returned from ACR_Open() function.	
[in] slotNum	Slot number.	
[in] controlCode	Control code for the operation. This value identifies the specific operation to be performed.	
[in] inBuffer	Pointer to a buffer that contains the data required to perform the operation. This parameter can be NULL if the controlCode parameter specifies an operation that does not require input data.	
[in] inBufferSize	Size, in bytes, of the buffer pointed to by inBuffer.	
[out] outBuffer	Pointer to a buffer that receives the operation's output data. This parameter can be NULL if the controlCode parameter specifies an operation that does not produce output data.	
[in] outBufferSize	Size, in bytes, of the buffer pointed to by outBuffer.	
[out] pBytesReturned	Pointer to a DWORD that receives the size, in bytes, of the data stored into the buffer pointer to by outBuffer.	
Return Value	ERROR_SUCCESS	The operation completed successfully.
	Failure	An error code. For Windows Platform, see error codes from Windows API and ACR error codes. For Linux Platform, see error codes from GNU C library header file "errno.h" and ACR error codes.

Table 14: ACR_Control Function Description

Source Code Example:

```
BYTE command[] = { 0xE0, 0x00, 0x00, 0x18, 0x00 };
DWORD commandLen = sizeof(command);
BYTE response[20];
DWORD responseLen = sizeof(response);
DWORD ret;

// Get firmware version
ret   =   ACR_Control(hReader,   0,   IOCTL_ACR_CCID_ESCAPE,   command,
commandLen, response, responseLen, &responseLen);
if (ret != ERROR_SUCCESS)
   printf("ACR_Control failed with error 0x%08X\n", ret);
```

Appendix A. Data Structures

Appendix A.1. _ACR_TIMEOUTS Struct Reference

This data structure is used in `ACR_GetTimeouts()` and `ACR_SetTimeouts()` function.

1. `DWORD statusTimeout`
   - Status timeout in milliseconds
   - Default is 2000 ms
2. `DWORD numStatusRetries`
   - Number of status retries
   - Default is 1
3. `DWORD responseTimeout`
   - Response timeout in milliseconds
   - Default is 10000 ms
4. `DWORD numResponseRetries`
   - Number of response retries
   - Default is 1

Appendix B. acr.h File Reference

Appendix B.1. Data Structure

1. `struct` `_ACR_TIMEOUTS`

   Timeouts

Appendix B.2. Defines

1. `#define` `ACR_ERROR_PROTO_MISMATCH ((DWORD) 0xA010000FL)`

   The requested protocols are incompatible with the protocol currently in use with the card.

2. `#define` `ACR_ERROR_UNSUPPORTED_CARD ((DWORD) 0xA0100065L)`

   The reader cannot communicate with the card, due to ATR string configuration conflicts.

3. `#define` `ACR_ERROR_UNRESPONSIVE_CARD ((DWORD) 0xA0100066L)`

   The smart card is not responding to a reset.

4. `#define` `ACR_ERROR_UNPOWERED_CARD ((DWORD) 0xA0100067L)`

   Power has been removed from the smart card, so that further communication is not possible.

5. `#define` `ACR_ERROR_REMOVED_CARD ((DWORD) 0xA0100069L)`

   The smart card has been removed, so further communication is not possible.

6. `#define` `ACR_PROTOCOL_UNDEFINED 0X00000000`

   Protocol is undefined.

7. `#define` `ACR_PROTOCOL_T0 0X00000001`

   T=0 protocol.

8. `#define` `ACR_PROTOCOL_T1 0X00000002`

   T=1 protocol.

9. `#define` `ACR_ICC_UNKNOWN 0`

   The current status is unknown.

10. `#define` `ACR_ICC_ABSENT 1`

    There is no card in the reader.

11. `#define` `ACR_ICC_PRESENT 2`

    There is a card in the reader.

12. `#define` `ACR_ICC_POWERED 3`

    Power is being provided to the card.

13. `#define` `ICOTL_ACR_CCID_ESCAPE 3500`

    Control code for sending CCID escape command.

14. `#define` `ACR_Open ACR_OpenA`

    `ACR_Open` will be mapped to `ACR_OpenW()` function of UNICODE is defined.

Appendix B.3. Typedefs

1.	`typedef struct _ACR_TIMEOUTS`	`ACR_TIMEOUTS` Create a type name for `_ACR_TIMEOUTS`.
2.	`typedef struct _ACR_TIMEOUTS*`	`ACR_TIMEOUTS` Create a type name for pointer to `_ACR_TIMEOUTS` data structure.