Embedded API (EAPI)
Developer Guide
V1.4

Advantech IIoT Group

July 11, 2018

Table of Contents

Revision History

Revision	Date	Comment
1.4	2018/07/11	Add netmask for network functions
1.3	2018/07	Add network functions
1.2	2018/01	Fixed typo
RC 1.1	2017/12	Add Return Status Code table for EApiETPReadUserData
RC 1.0	2017/10	Revision Candidate 1.0

1. Introduction

Embedded API (EAPI) follows PICMG EAPI to specify functions for industrial application and provide a common programming interface. The target is to avoid software modifications when changing device modules. EAPI will cover all interfaces in the device to unify the software control for:

- System information
- UPS
- ETP
- GPIO
- NVRAM (User storage area)
- Electric type plate
- HW monitor
- Brightness
- EC extension functions
- Network functions

2. General

2.1. Parameters

Parameters which can return values are defined as pointers to the data. The other parameters are defined as values. The immediate return value is an error code.

2.2. Keywords

In order to improve the readability, this document features keywords used before variables.

2.2.1 __IN

Parameter Type	Characteristics
Immediate value	Input value that must be specified and is essential
Pointer	Valid pointer to initialized buffer/variable.

2.2.2 __OUT

Parameter Type	Characteristics
Pointer	Valid pointer to initialized buffer/variable.

2.2.3 __INOPT

Parameter Type	Characteristics
Pointer	Valid pointer to initialized buffer/variable, or NULL Pointer. Note: refer to function specification for specifics.

2.2.4 __OUTOPT

Parameter Type	Characteristics
Pointer	Valid pointer to initialized buffer/variable, or NULL Pointer. Note: refer to function specification for specifics.

2.2.5 __INOUT

Parameter Type	Characteristics
Pointer	A valid pointer to initialized buffer/variable. Contents of buffer/variable updated before return.

2.3. Status Codes

EAPI_STATUS_SUCCESS

Value

0x0

Description

The operation was successful.

Actions

None.

EAPI_STATUS_ERROR

Value

0xFFFFF0FF

Description

Generic error message. No further error details are available.

Actions

None.

EAPI_STATUS_GET_STATUS_ERROR

Value

0xFFFFF8FF

Description

Failed to get the status.

Actions

None.

EAPI_STATUS_LOCKED

Value

0xFFFFF8FE

Description

The storage is locked and read-only.

Actions

Unlock first and retry.

EAPI_STATUS_MORE_DATA

Value

0xFFFFF9FF

Description

The amount of available data exceeds the buffer size.

Storage buffer overflow was prevented. Read count was larger than the defined buffer length.

Actions

Either increase the buffer size or reduce the block length.

EAPI_STATUS_WRITE_ERROR

Value

0xFFFFFAFE

Description

An error was detected during a write operation.

Example

I2C Write function was not successful.

No Acknowledge was received after writing any byte after the first address byte.

Can be caused by unsupported device command/index.

10Bit Address Device Not Present

Storage Write Error

Actions

Retry.

EAPI_STATUS_READ_ERROR

Value

0xFFFFFAFF

Description

An error was detected during a read operation.

Example

The I2C Read function was not successful

Actions

Retry.

EAPI_STATUS_TIMEOUT

Value

0xFFFFFBFE

Description

Function call timed out

Example

The I2C operation lasted too long.

Actions

Retry.

EAPI_STATUS_RUNNING

Value

0xFFFFFEFA

Description

The function already started.

Actions

None.

EAPI_STATUS_BUSY_COLLISION

Value

0xFFFFFBFD

Description

The selected device or ID is busy, or a data collision was detected.

Example

The addressed I2C bus is busy, or there is a bus collision.

The I2C bus is in use. Either CLK or DAT is low.

Arbitration loss or bus Collision, data remains low when writing a 1.

Actions

Retry.

EAPI_STATUS_NOT_FOUND

Value

0xFFFFFBFF

Description

The selected device was not found.

Actions

None.

EAPI_STATUS_UNSUPPORTED

Value

0xFFFFFCFF

Description

This function or ID is not supported in the platform environment.

Actions

None.

EAPI_STATUS_INVALID_PARAMETER

Value

0xFFFFFEFF

Description

One or more of the EAPI function call parameters are out of the defined range.

Actions

Verify Function Parameters.

EAPI_STATUS_INVALID_BLOCK_LENGTH

Value

0xFFFFFEFD

Description

This status means that the Block length is too long.

Actions

Use relevant Capabilities information to correct select block lengths.

EAPI_STATUS_INVALID_BLOCK_ALIGNMENT

Value

0xFFFFFEFE

Description

The Block Alignment is incorrect.

Actions

Use Alignment Capabilities information to align write access correctly.

EAPI_STATUS_INVALID_DIRECTION

Value

0xFFFFFEFC

Description

The current Direction Argument attempts to set GPIOs to unsupported directions. I.E., Setting GPI to Output.

Actions

Use pInputs and pOutputs to select input and outputs correctly.

EAPI_STATUS_NOT_INITIALIZED

Value

0xFFFFFFFF

Description

The EAPI library is not yet or unsuccessfully initialized. EApiLibInitialize needs to be called before the first access of any other EAPI function.

Actions

Call EApiLibInitialize.

EAPI_STATUS_INITIALIZED

Value

0xFFFFFFFE

Description

The library is initialized.

Actions

None.

EAPI_STATUS_ALLOC_ERROR

Value

0xFFFFFFFD

Description

Memory Allocation Error.

Actions

Free memory and try again.

3. Information Functions

```
EApiStatus_t EApiBoardGetStringA(
    __IN    EApiId_t   Id,
    __OUT   char*      pBuffer,
    __INOUT uint32_t* pBufLen
    );
```

Description:

Text information about the hardware platform.

Parameters

Id

__IN Selects the Get String Sub-function Id.

EAPI_ID_BOARD_NAME_STR	0x1
EAPI_ID_BOARD_BIOS_REVISION_STR	0x4
EAPI_ID_BOARD_EC_REVISION_STR	0x101
EAPI_ID_BOARD_OS_REVISION_STR	0x102
EAPI_ID_BOARD_CPU_MODEL_NAME_STR	0x103

pBuffer

__OUT Pointer to a buffer that receives the value's data.

pBufLen

__IN Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pBuffer parameter.

Return Status Code

EAPI_STATUS_ERROR	Open kernel device error
EAPI_STATUS_ALLOC_ERROR	Create buffer error
EAPI_STATUS_SUCCESS	Success

```
EApiStatus_t EApiBoardGetValue (
    __IN    EApiId_t   Id,
    __OUT   uint32_t*  pValue
    );
```

Description:

Information about the hardware platform in value format.

Parameters

Id

__IN Selects the Get Value Sub function Id.

EAPI_ID_HWMON_TEMP_CPU	CPU Temperature
EAPI_ID_HWMON_TEMP_SYSTEM	System Temperature

EAPI_ID_HWMON_VOLTAGE_VCORE	CPU Core Voltage

EAPI_ID_HWMON_VOLTAGE_VCORE2	CPU Core Voltage
EAPI_ID_HWMON_VOLTAGE_2V5	2.5V Voltage
EAPI_ID_HWMON_VOLTAGE_3V3	3.3V Voltage
EAPI_ID_HWMON_VOLTAGE_5V	5V Voltage
EAPI_ID_HWMON_VOLTAGE_12V	12V Voltage
EAPI_ID_HWMON_VOLTAGE_5VSB	5V Standby Voltage
EAPI_ID_HWMON_VOLTAGE_3VSB	3V Standby Voltage
EAPI_ID_HWMON_VOLTAGE_VBAT	Battery Voltage
EAPI_ID_HWMON_VOLTAGE_5NV	-5V Voltage
EAPI_ID_HWMON_VOLTAGE_12NV	-12V Voltage
EAPI_ID_HWMON_VOLTAGE_VTT	DIMM Voltage
EAPI_ID_HWMON_VOLTAGE_24V	24V Voltage
EAPI_ID_HWMON_VOLTAGE_DC	DC Input Voltage
EAPI_ID_HWMON_VOLTAGE_DCSTBY	DC Standby Voltage
EAPI_ID_HWMON_VOLTAGE_VBATLI	Li-ion Battery Voltage
EAPI_ID_HWMON_VOLTAGE_OEM0~3	Other Voltages
EAPI_ID_HWMON_VOLTAGE_1V05	1.05V Voltage
EAPI_ID_HWMON_VOLTAGE_1V5	1.5V Voltage
EAPI_ID_HWMON_VOLTAGE_1V8	1.8V Voltage

pBuffer

__OUT Pointer to a buffer that receives the value's data.

pBufLen

__IN Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pBuffer parameter.

Return Status Code

EAPI_STATUS_ERROR	Open kernel device error
EAPI_STATUS_ALLOC_ERROR	Create buffer error
EAPI_STATUS_SUCCESS	Success

```
EApiStatus_t EApiGetCOMports(
    __INOUT     PLATFORM_COMPORT**          comports,
    __INOUT     UINT*                       len
);
```

Description:

Get device COM ports array. Each COM port will be added to the PLATFORM_COMPORT structure.

Parameters

comports

__INOUT The array of the com ports will be returned.

len

__INOUT the length(size) of the array.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The input of comports is not NULL.
EAPI_STATUS_ALLOC_ERROR	Create buffer error
EAPI_STATUS_SUCCESS	Success

```
EApiStatus_t EApiGetMemoryAvailable(
    __INOUT     float          *mem_available
);
```

Description:

Get device available memory usage (KB).

Parameters

mem_available

__INOUT The point of float number.

Return Status Code

EAPI_STATUS_SUCCESS	Success

```
EApiStatus_t EApiGetDiskInfo(
     INOUT     PDISK INFO       diskInfo
);
```

Description:

Get device Disk information. Disk partition information will be added to `DISK_INFO` structure.

Parameters

diskInfo

__INOUT The disk partition structure will be returned.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The input of diskInfo is not NULL.
EAPI_STATUS_ALLOC_ERROR	Memory Allocation Error
EAPI_STATUS_SUCCESS	Success

4. UPS Functions

```
EApiStatus_t EApiUPSGetDevice(
      INOUT EApiCommonDLL::atIPSAE** device,
    __IN    string                   port,
    );
```

Description:

Create UPS Object.

Parameters

device

__INOUT the point of UPS class object

port

__IN The COM port name string. Ex. COM1

Return

EApiCommonDLL::IatIPSAE* class Object.

```
EApiStatus_t EApiUPSDelDevice(
    void
    );
```

Description:

Delete UPS Object. You need EApiUPSDelDevice after you did not use ups anymore.

Parameters

None

5. ETP Functions

```
EApiStatus_t EApiGetLockStatus (
      IN       int SalveAddr
    __INOUT     DWORD * pLockStatus

    );
```

Description:

Get the area lock status.

Parameters

`SalveAddr`

__ IN the area address of EEPROM location.

`LockStatus`

__ INOUT the pointer to store lock status.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The pLockStatus is NULL.
EAPI_STATUS_NOT_FOUND	The driver not found.
EAPI_STATUS_ERROR	Failed to call the driver, call **GetLastError** to get the detail error code.
EAPI_STATUS_SUCCESS	Success.

```
EApiStatus_t EApiSetEepromProtect(
     IN       int SalveAddr
    __IN      BOOL bProtect
    __IN      UCHAR *pPassword
    __IN      int PasswordLen
    );
```

Description:

Lock/Unlock the area.

Parameters

`SalveAddr`

__ IN the area address of EEPROM location.

`bProtect`

__ IN the value to lock/unlock area.

`pPassword`

__ IN the password to lock/unlock area.

`PasswordLen`

__ IN the Password length, the maximum length is 8.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The pPassword is NULL, Incorrect PasswordLen,
EAPI_STATUS_ALLOC_ERROR	Failed to allocate a buffer.
EAPI_STATUS_NOT_FOUND	The driver not found.
EAPI_STATUS_ERROR	Failed to call the driver, call **GetLastError** to get the detail error code.
EAPI_STATUS_SUCCESS	Success.

```
EApiStatus_t EApiETPReadDeviceData(
    __INOUT    PETP_DATA pOutBuffer
    );
```

Description:

Get the device information with structure PETP_DATA. The structure is :

```
typedef struct {
     UCHAR DeviceOrderText[40];          // A4, offset 0
     UCHAR DeviceOrderNumber[10];  // A4, offset 40
     UCHAR DeviceIndex[3];               // A4, offset 50
     UCHAR DeviceSerialNumber[15]; // A4, offset 53
     UCHAR OperatingSystem[40];          // A4, offset 68
     UCHAR Image[40];                    // A4, offset 108
     UCHAR Reverse[92];
} ETP_DATA, *PETP_DATA;
```

Parameters

`pOutBuffer`

__ INOUT get the device area information.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The pOutBuffer is NULL.
EAPI_STATUS_NOT_FOUND	The driver not found.

EAPI_STATUS_ERROR	Failed to call the driver, call **GetLastError** to get the detail error code.
EAPI_STATUS_SUCCESS	Success.

```
EApiStatus_t EApiETPReadUserData (
    __INOUT    PETP_USER_DATA pOutBuffer
    );
```

Description:

Get the device information with structure PETP_USER_DATA. The structure is :

```
typedef struct {
     UCHAR UserSpace1[128];              // A6, offset 0
     UCHAR UserSpace2[128];              // A6, offset 128
} ETP_USER_DATA, *PETP_USER_DATA;
```

Parameters

pOutBuffer

__ INOUT get the user area information.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The pOutBuffer is NULL.
EAPI_STATUS_NOT_FOUND	The driver not found.
EAPI_STATUS_ERROR	Failed to call the driver, call **GetLastError** to get the detail error code.
EAPI_STATUS_SUCCESS	Success.

```
EApiStatus_t EApiETPWriteUserData (
    __INOUT    PETP_USER_DATA pOutBuffer
    );
```

Description:

Set the device information with structure PETP_USER_DATA. The structure is :

```
typedef struct {
     UCHAR UserSpace1[128];              // A6, offset 0
     UCHAR UserSpace2[128];              // A6, offset 128
} ETP_USER_DATA, *PETP_USER_DATA;
```

Parameters

pOutBuffer

__ INOUT the user area information you want to set.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	The pOutBuffer is NULL.
EAPI_STATUS_NOT_FOUND	The driver not found.
EAPI_STATUS_ERROR	Failed to call the driver, call **GetLastError** to get the detail error code.
EAPI_STATUS_SUCCESS	Success.
EAPI_STATUS_GET_STATUS_ERROR	Get lock status error.

EAPI_STATUS_LOCKED	The area is locked.

6. GPIO Functions

GPIO ID:

EAPI_GPIO_GPIO_ID(GPIO_NUM)	GPIO_NUM is GPIO pin number. (Single pin only)
EAPI_ID_GPIO_BANK(BANK_NUM)	BANK_NUM is GPIO bank number(Maximum 32 pins per bank)

```
EApiStatus_t EApiGPIOGetLevel (
    __IN       EApiId_t  Id,
    __IN       uint32_t  Bitmask,
    __OUT      uint32_t *pLevel
    );
```

Description:

Get GPIO level of selected GPIO pin(s).

Parameters:

Id

__IN GPIO Id.

Bitmask

__IN Bit mask of Affected Bits.

pLevel

__OUT Pointer to a buffer receives the Current Level.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get GPIO level.
EAPI_STATUS_INVALID_PARAMETER	pLevel is NULL. Bitmask is 0 Incorrect GPIO Id
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvGPIO driver handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiGPIOSetLevel (
     IN       EApiId t  Id,
     IN       uint32 t  Bitmask,
    __IN      uint32_t  Level
    );
```

Description:

Set GPIO level of selected GPIO pin(s).

Parameters:

Id

__IN GPIO Id.

Bitmask

__IN Bit mask of Affected Bits.

Level

__IN New level of selected GPIO pin(s). High (1), Low (0)

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to set GPIO level.
EAPI_STATUS_INVALID_PARAMETER	Bitmask is 0 Incorrect GPIO Id
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvGPIO driver handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiGPIOGetDirection (
      IN       EApiId_t  Id,
      IN       uint32_t  Bitmask,
    __OUT      uint32_t *pDirection
    );
```

Description:

Get direction of selected GPIO pin(s).

Parameters:

Id

__IN GPIO Id.

Bitmask

__IN Bit mask of Affected Bits.

pDirection

__OUT Pointer to a buffer that receives the GPIO Direction.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get GPIO direction.
EAPI_STATUS_INVALID_PARAMETER	pDirection is NULL. Bitmask is 0 Incorrect GPIO Id
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvGPIO driver handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiGPIOSetDirection (
    __IN       EApiId_t  Id,
      IN       uint32_t  Bitmask,
      IN       uint32_t  Direction
    );
```

Description:

Set selected GPIO pin(s) to input or output direction.

Parameters:

Id

__IN GPIO Id.

Bitmask

__IN Bit mask of Affected Bits.

Direction

__IN New direction for the selected GPIO pin(s).

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to set GPIO direction.
EAPI_STATUS_INVALID_PARAMETER	Bitmask is 0 Incorrect GPIO Id
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvGPIO driver handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiGPIOGetDirectionCaps (
     IN       EApiId t  Id,
     OUTOPT   uint32 t *pInputs,
    __OUTOPT  uint32_t *pOutputs
   );
```

Description:

Get GPIO direction capability.

Parameters:

Id

__IN GPIO Id.

pInputs

__OUTOPT Pointer to a buffer that receives the supported GPIO input bit mask.

pOutputs

__OUTOPT Pointer to a buffer that receives the supported GPIO output bit mask.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get GPIO direction.
EAPI_STATUS_INVALID_PARAMETER	Both pInputs and pOutputs are NULL. Incorrect GPIO Id
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvGPIO driver handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

7. Brightness Functions

Backlight ID:

EAPI_ID_BACKLIGHT_1	The main backlight control.

```
EApiStatus_t EApiDisplayGetBacklightBrightness (
     IN       EApiId t  Id,
     OUT      uint32 t *pBright
   );
```

Description:

Get the brightness value.

Parameters:

Id

__IN Backlight Id.

pBright

__OUT Pointer to the brightness value.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get the brightness value.
EAPI_STATUS_INVALID_PARAMETER	pBright is NULL.
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvBrightness driver handle. Incorrect backlight Id.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiDisplaySetBacklightBrightness (
   __IN      EApiId_t  Id,
   __IN      uint32_t  Bright
   );
```

Description:

Set the brightness value.

Parameters:

Id

__IN Backlight Id.

Bright

__IN The new brightnees value.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to set a new brightness value.
EAPI_STATUS_INVALID_PARAMETER	Bright > brightness maximum value Bright < brightness minimum value.
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvBrightness driver handle. Incorrect backlight Id.

EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiDisplayGetCap (
     IN       EApiId_t  Id,
     IN       uint32_t  CapId,
    __OUT     uint32_t *pValue
    );
```

Description:

Get display capabilities.

Parameters:

Id

__IN Panel/Backlight Id.

CapId

__IN capability Id.

pValue

__OUT Pointer to the capability Id.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get the value.
EAPI_STATUS_INVALID_PARAMETER	pValue is NULL
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvBrightness driver handle. Incorrect backlight Id. Incorrect capability Id or not support.
EAPI_STATUS_ERROR	Failed to call IOCTL.

Capability ID:

EAPI_ID_DISPLAY_BRIGHTNESS_MAXIMUM	The maximum brightness value of the device.
EAPI_ID_DISPLAY_BRIGHTNESS_MINIMUM	The minimum brightness value of the device.
EAPI_ID_DISPLAY_AUTO_BRIGHTNESS	On/Off status of Auto-Brightness function

```
EApiStatus_t EApiDisplaySetCap (
     IN       EApiId_t  Id,
     IN       uint32_t  CapId,
    __IN      uint32_t  Value
    );
```

Description:

Set a new capability value.

Parameters:

Id

__IN Panel/Backlight Id.

CapId

__IN capability Id.

Value

__IN New capability value.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get the value.
EAPI_STATUS_INVALID_PARAMETER	If *CapId* == EAPI_ID_DISPLAY_AUTO_BRIGHTNESS, *Value* != EAPI_AUTO_BRIGHTNESS_SET_ON (1) or *Value* != EAPI_AUTO_BRIGHTNESS_SET_OFF (0)
EAPI_STATUS_UNSUPPORTED	Failed to open the AdvBrightness driver handle. Incorrect backlight Id. Incorrect capability Id or not support.
EAPI_STATUS_ERROR	Failed to call IOCTL.

Capability ID:

EAPI_ID_DISPLAY_AUTO_BRIGHTNESS	On/Off status of Auto-Brightness function

8. NVRAM Disk Functions

```
EApiStatus_t EApiNVRAMDiskCap(
      IN       TCHAR *pDiskLabel,
    __INOUT    ULONG *puStorageSize
    );
```

Description:

Get the NVRAM disk size (Unit: Bytes).

Parameters:

pDiskLabel

__IN Pointer to the disk label.

puStorageSize

__INOUT Pointer to the storage size.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get the NVRAM disk size.
EAPI_STATUS_INVALID_PARAMETER	puStorageSize is NULL.
EAPI_STATUS_UNSUPPORTED	Failed to open the NVRAMDisk handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiNVRAMDiskGetStatus(
    __IN       TCHAR *pDiskLabel,
    __INOUT    UCHAR *puStatus
    );
```

Description:

Get the status of direct access to the disk.

Parameters:

pDiskLabel

__IN Pointer to the disk label.

puStatus

__INOUT Pointer to the status of direct access to the disk.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get the status of direct access to the disk.
EAPI_STATUS_INVALID_PARAMETER	puStatus is NULL.
EAPI_STATUS_UNSUPPORTED	Failed to open the NVRAMDisk handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiNVRAMDiskSetStatus(
      IN       TCHAR *pDiskLabel,
      IN       UCHAR uStatus
    );
```

Description:

Set the status of direct access to the disk.

Parameters:

pDiskLabel

__IN Pointer to the disk label.

uStatus

__IN Enable or disable the direct access to the disk mode.

Value	Description
1	Enable the direct access to the disk mode.
0	Disable the direct access to the disk mode.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to set the status of direct access to the disk.
EAPI_STATUS_UNSUPPORTED	Failed to open the NVRAMDisk handle.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiNVRAMDiskReadRaw(
      IN       TCHAR *pDiskLabel,
    __IN       ULONG uOffset,
    __INOUT    void *pBuffer,
      IN       ULONG uBufLen,
      IN       ULONG uByteCnt
    );
```

Description:

Read the raw data from the NVRAM Disk.

Parameters:

pDiskLabel

__IN Pointer to the disk label.

uOffset

__IN The data offset in bytes.

pBuffer

__INOUT Pointer to the data buffer.

uBufLen

__IN The data buffer size in bytes.

uByteCnt

__IN The number of bytes to read.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to read the raw data from the NVRAM disk.
EAPI_STATUS_INVALID_PARAMETER	pBuffer is NULL.
EAPI_STATUS_MORE_DATA	Read Count is larger than Buffer Length.
EAPI_STATUS_UNSUPPORTED	Failed to open the NVRAMDisk handle.
EAPI_STATUS_ALLOC_ERROR	Failed to allocate a buffer.
EAPI_STATUS_ERROR	Failed to call IOCTL.

```
EApiStatus_t EApiNVRAMDiskWriteRaw(
      IN       TCHAR *pDiskLabel,
      IN       ULONG uOffset,
      IN       void *pBuffer,
    __IN       ULONG uBufLen,
```

```
    IN      ULONG uByteCnt
);
```

Description:

Write the raw data to the NVRAM Disk.

Parameters:

pDiskLabel

__IN Pointer to the disk label.

uOffset

__IN The data offset in bytes.

pBuffer

__INOUT Pointer to the data buffer.

uBufLen

__IN The data buffer size in bytes.

uByteCnt

__IN The number of bytes to write.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to write the raw data from the NVRAM disk.
EAPI_STATUS_INVALID_PARAMETER	pBuffer is NULL.
EAPI_STATUS_MORE_DATA	Write Count is larger than Buffer Length.
EAPI_STATUS_UNSUPPORTED	Failed to open the NVRAMDisk handle.
EAPI_STATUS_ALLOC_ERROR	Failed to allocate a buffer.
EAPI_STATUS_ERROR	Failed to call IOCTL.

9. EC Extension Functions

Macro	Description
EAPI_EXT_FUNC_LED_MAX	The maximum number LED supported by the EC firmware which is 16.
EAPI_ID_EXT_FUNC_LED_BASE	0x00000000
EAPI_ID_EXT_FUNC_LED(N)	(0x00000000 + N), N: 0~15 0x00000000 ~ 0x0000000f
EAPI_ID_EXT_FUNC_LED_MIN	0x00000000
EAPI_ID_EXT_FUNC_LED_MAX	0x0000000f
EAPI_EXT_FUNC_LED_ID_TO_INDEX(ID)	Convert LED function Id to Index
EAPI_EXT_FUNC_POWER_VIN_MAX	The maximum Power Status supported by the EC firmware which is 2.
EAPI_ID_EXT_FUNC_POWER_STATUS_BASE	0x00000010
EAPI_ID_EXT_FUNC_POWER_STATUS_VIN(N)	(0x00000010 + N) , N: 0~1 0x00000010 ~ 0x00000011
EAPI_ID_EXT_FUNC_POWER_STATUS_VIN_MIN	0x00000010
EAPI_ID_EXT_FUNC_POWER_STATUS_VIN_MAX	0x00000011
EAPI_EXT_FUNC_POWER_STATUS_VIN_ID_TO_INDEX	Convert Power Status function Id to Index
EAPI_ID_EXT_FUNC_MAX	0x000000ff; The maximum function Id

```
EApiStatus_t EApiExtFunctionGetStatus (
    __IN    EApiId_t   Id,
    __INOUT  uint32_t* pStatus
   );
```

Description:

Get the status of the specified function.

Parameters

Id

__IN Selects the EC Extension function Id.

EAPI_ID_EXT_FUNC_ LED(N)	N: 0~15
EAPI_ID_EXT_FUNC_POWER_STATUS_VIN(N)	N: 0~1

pStatus

__INOUT Pointer to a buffer that receives the status data.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	Invalid *Id* or *pStatus* is NULL.
EAPI_STATUS_UNSUPPORTED	The platform does not support this function.
EAPI_STATUS_ERROR	Call GetLastError() to get the detail error code.

EAPI_STATUS_SUCCESS	Success

```
EApiStatus_t EApiExtFunctionSetStatus (
    __IN    EApiId_t   Id,
      IN    uint32_t   Status
    );
```

Description:

Set a new status to the specified function.

Parameters

Id

__IN Selects the EC Extension function Id.

EAPI_ID_EXT_FUNC_ LED(N)	N: 0~15
EAPI_ID_EXT_FUNC_POWER_STATUS_VIN(N)	N: 0~1

Status

__IN Pointer to a buffer that receives the status data.

Return Status Code

EAPI_STATUS_INVALID_PARAMETER	Invalid *Id* or the *Status* is greater than 0xFF.
EAPI_STATUS_UNSUPPORTED	The platform does not support this function.
EAPI_STATUS_ERROR	Call GetLastError() to get the detail error code.
EAPI_STATUS_SUCCESS	Success

10. Network Functions

```
enum IP FAMILY
{
    kIPv4 = 2, // AF_INET
    kIPv6 = 23 // AF_INET6
};
```

Description:

The **IP_FAMILY** enumeration specifies the IP family type.

Constants:

kIPv4	IPv4
kIPv6	IPv6

```
typedef struct  IP ADDRESS
{
   struct  IP ADDRESS *next;
   unsigned short family;
   char *address;
} IP_ADDRESS, *PIP_ADDRESS;
```

Description:

The **IP_ADDRESS** structure stores an IP address in a linked list of IP addresses for a particular adapter.

Members:

next	A pointer to the next IP address structure in the list.
family	kIPv4 for IPv4, kIPv6 for IPv6
address	IP address

```
typedef struct  STRING LIST
{
   struct _STRING_LIST *next;
   char *string;
} STRING LIST, *PSTRING LIST;
```

Description:

The **STRING_LIST** structure stores a string in a linked list of strings.

Members:

next	A pointer to the next string structure in the list.
string	string

```
typedef struct  CONN CONTEXT
{
    struct _CONN_CONTEXT *next;
    char *access_string;
    char *user_name;
    char *password;
    bool compression;
    char *auth type;
} CONN_CONTEXT, *PCONN_CONTEXT;
```

Description:

The **CONN_CONTEXT** structure stores a Mobile Broadband Network connection context in a linked list of connection contexts.

Members:

next	A pointer to the next connection context structure in the list.
access_string	Contains connection-specific access information. In GSM networks, this would be an access point name (APN) such as "data.thephone-company.com". In CDMA networks, this might be a special dial code such as "#777" or a NAI (Network Access Identifier) such as "somebody@thephone-company.com".
user_name	Contains the user name that is used for authentication.
password	Contains the password that is used for authentication.
compression	Specifies whether compression is to be used in the data link for header and data. This member is applicable only for GSM devices.
auth_type	Indicates the type of compression used for PDP (Packet Data Protocol) activation. It could be "NONE", "PAP", "CHAP", "MsCHAPv2", or "*** UNKNOWN ***"

```
// WiFi
typedef struct  WIFI INFO
{
    bool is_connected;
    char *profile;
    char *ssid;
    char *bssid;
    int bss type;
    int signal_strength; // dBm
    int signal_quality;  // 0~100 %
    bool security_enabled;
    bool onex enabled;
    unsigned long rx rate;
    unsigned long tx rate;
    char *auth_algorithm;
    char *cipher_algorithm;
    void *reserved;
} WIFI INFO, *PWIFI INFO;
```

Description:

The **WIFI_INFO** structure stores Wi-Fi information.

Members:

is_connected	Indicates whether interface is connected to a network.
profile	The name of the profile used for the connection.
ssid	The SSID of the association.
bss_type	Basic service set (BSS) network type. 1 -> infrastructure BSS network, 2 -> independent BSS (IBSS) network (ad hoc) 3 -> either infrastructure or IBSS network.
signal_strength	Actual RSSI signal strength (dbm)
signal_quality	A percentage value that represents the signal quality of the network.
security_enabled	Indicates whether security is enabled for this connection.
onex_enabled	Indicates whether 802.1X is enabled for this connection.
rx_rate	The receiving rate of the association.
tx_rate	The transmission rate of the association.
auth_algorithm	The authentication algorithm. It could be "802.11 Open", "802.11 Shared", "WPA", "WPA-PSK", "WPA-None", "RSNA", or "RSNA with PSK".
cipher_algorithm	The cipher algorithm. It could be "None", "WEP-40", "TKIP", "CCMP", "WEP-104", or "WEP".
reserved	Reserved for future use.

```
// Mobile Broadband Network
typedef struct  MBN_INFO
{
    bool is_connected;
    char *profile;
    char *imei;
    char *manufacturer;
    char *model;
    char *firmware;
    char *cellular_class;
    char *band_class;
    char *sim iccid;
    char *subscriber id; // GSM -> IMSI, CDMA -> MIN
    char *data classes;
    char *current_data_class;
    int signal_strength; // dBm
    int signal quality;  // 0~100 %
    PCONN CONTEXT conn ctx list;
    char *provider name;
    PSTRING_LIST phone_numbers;
    void *reserved;
} MBN_INFO, *PMBN_INFO;
```

Description:

The **MBN_INFO** structure stores a Mobile Broadband Network information.

Members:

is_connected	Indicates whether interface is connected to a network.
profile	The name of the profile used for the connection.
imei	IMEI (up to 15 digits) for GSM devices or ESN (11 digits) / MEID (17 digits) for CDMA devices.
manufacturer	The name of the device manufacturer. It can be empty.
model	The device model. It can be empty.
firmware	The firmware-specific information for this device. It can be empty.
cellular_class	The type of cellular device. It could be "NONE", "GSM", "CDMA", or "*** UNKNOWN ***".
band_class	The frequency band classes.
sim_iccid	The SIM International circuit card number (SimICCID) for the device.
subscriber_id	The subscriber ID of the device. For GSM device this represents the International Mobile Equipment Identity (IMSI) string (up to 15 digits). For CDMA device this represents the Mobile Identification Number (MIN) string or the International Roaming MIN (IRM) string (10 digits).
data_classes	Specifies which data services are supported. For GSM devices, only the GSM-based data services can be present, that is, only GPRS, EDGE, UMTS, LTE, and HSDPA are valid values for GSM devices. For CDMA devices, only the CDMA-related data services will

	be present, that is, only 1xRTT, 1xEV-DO, and 1xEV-DO RevA are valid values for CDMA devices. 1xEV-DO RevB is reserved for future use.
current_data_class	The current data class in the current network.
signal_strength	Actual RSSI signal strength (dbm)
signal_quality	A percentage value that represents the signal quality of the network.
conn_ctx_list	A list of connection contexts.
provider_name	The provider name for the currently registered network.
phone_numbers	The telephone numbers associated with the device.
reserved	Reserved for future use.

```
typedef struct  NETWORK INFO
{
    struct _NETWORK_INFO *next;
    int index;
    unsigned long mtu;
    unsigned long flags;
    unsigned long iftype;
    unsigned long operstatus;
    bool connected;
    bool is_default;
    bool dhcp_enabled;
    char *uuid;
    char *name;
    char *description;
    char *mac_address;
    PIP_ADDRESS addresses;
    PIP ADDRESS gateways;
    PIP ADDRESS dns;
    PIP ADDRESS ipv4 netmask;
    PWIFI_INFO wifi_info;
    PMBN_INFO mobile_broadband_info;

    void *reserved;
} NETWORK_INFO, *PNETWORK_INFO;
```

Description:

The **NETWORK_INFO** is the *header node* for a linked list of network information for a particular adapter. This structure can simultaneously be used as part of a linked list of NETWORK_INFO structures.

Members:

<table>
<tr><td>next</td><td>A pointer to the next network info structure in the list.</td></tr>
<tr><td>index</td><td>The interface index.</td></tr>
<tr><td>mtu</td><td>The maximum transmission unit (MTU) size, in bytes.</td></tr>
<tr><td>flags</td><td>A set of flags specifying various settings for the adapter.
<table>
<tr><td>IP_ADAPTER_DDNS_ENABLED</td><td>0x00000001</td></tr>
<tr><td>IP_ADAPTER_REGISTER_ADAPTER_SUFFIX</td><td>0x00000002</td></tr>
<tr><td>IP_ADAPTER_DHCP_ENABLED</td><td>0x00000004</td></tr>
<tr><td>IP_ADAPTER_RECEIVE_ONLY</td><td>0x00000008</td></tr>
<tr><td>IP_ADAPTER_NO_MULTICAST</td><td>0x00000010</td></tr>
<tr><td>IP_ADAPTER_IPV6_OTHER_STATEFUL_CONFIG</td><td>0x00000020</td></tr>
<tr><td>IP_ADAPTER_NETBIOS_OVER_TCPIP_ENABLED</td><td>0x00000040</td></tr>
<tr><td>IP_ADAPTER_IPV4_ENABLED</td><td>0x00000080</td></tr>
<tr><td>IP_ADAPTER_IPV6_ENABLED</td><td>0x00000100</td></tr>
<tr><td>IP_ADAPTER_IPV6_MANAGE_ADDRESS_CONFIG</td><td>0x00000200</td></tr>
</table></td></tr>
<tr><td>Iftype</td><td>The interface type as defined by the Internet Assigned Names Authority (IANA). Possible values for the interface type are listed in the Ipifcons.h header file.
The table below lists common values for the interface type although many other values are possible.
<table>
<tr><th>Value</th><th>Meaning</th></tr>
</table></td></tr>
</table>

	IF_TYPE_OTHER 1	Some other type of network interface.
	IF_TYPE_ETHERNET_CSMACD 6	An Ethernet network interface.
	IF_TYPE_ISO88025_TOKENRING 9	A token ring network interface.
	IF_TYPE_PPP 23	A PPP network interface.
	IF_TYPE_SOFTWARE_LOOPBACK 24	A software loopback network interface.
	IF_TYPE_ATM 37	An ATM network interface.
	IF_TYPE_IEEE80211 71	An IEEE 802.11 wireless network interface. On Windows Vista and later, wireless network cards are reported as **IF_TYPE_IEEE80211**. On earlier versions of Windows, wireless network cards are reported as **IF_TYPE_ETHERNET_CSMACD**. On Windows XP with SP3 and on Windows XP with SP2 x86 with the Wireless LAN API for Windows XP with SP2 installed, the WlanEnumInterfaces function can be used to enumerate wireless interfaces on the local computer.
	IF_TYPE_TUNNEL 131	A tunnel type encapsulation network interface.
	IF_TYPE_IEEE1394	An IEEE 1394 (Firewire) high

	144 — performance serial bus network interface. **IF_TYPE_WWANPP** 243 — WWAN devices based on GSM technology. **IF_TYPE_WWANPP2** 244 — WWAN devices based on CDMA technology.
operstatus	The operational status for the interface as defined in RFC 2863. For more information, see http://www.ietf.org/rfc/rfc2863.txt. This member can be one of the values from the IF_OPER_STATUS enumeration type defined in the Iftypes.h header file. On Windows Vista and later, the header files were reorganized and this enumeration is defined in the Ifdef.h header file.

Value	Meaning
IfOperStatusUp 1	The interface is up and able to pass packets.
IfOperStatusDown 2	The interface is down and not in a condition to pass packets. The **IfOperStatusDown** state has two meanings, depending on the value of **AdminStatus** member. If **AdminStatus** is not set to **NET_IF_ADMIN_STATUS_DOWN** and **ifOperStatus** is set to **IfOperStatusDown** then a fault condition is presumed to exist on the interface. If **AdminStatus** is set to **IfOperStatusDown**, then **ifOperStatus** will normally also be set to **IfOperStatusDown** or **IfOperStatusNotPresent** and there is not necessarily a fault condition on the interface.
IfOperStatusTesting 3	The interface is in testing mode.
IfOperStatusUnknown 4	The operational status of the interface is unknown.
IfOperStatusDormant 5	The interface is not actually in a condition to pass packets (it is not up), but is in a pending state,

		waiting for some external event. For on-demand interfaces, this new state identifies the situation where the interface is waiting for events to place it in the **IfOperStatusUp** state.
	IfOperStatusNotPresent 6	A refinement on the **IfOperStatusDown** state which indicates that the relevant interface is down specifically because some component (typically, a hardware component) is not present in the managed system.
	IfOperStatusLowerLayerDown 7	A refinement on the **IfOperStatusDown** state. This new state indicates that this interface runs on top of one or more other interfaces and that this interface is down specifically because one or more of these lower-layer interfaces are down.
connected	Indicates whether adapter is connected to a network. (IfOperStatusUp)	
is_default	Indicates whether adapter is a default route.	
dhcp_enabled	Indicates whether DHCP is enabled on this adapter.	
uuid	The unique ID for the adapter.	
name	A user-friendly name for the adapter. For example: "Local Area Connection 1." This name appears in contexts such as the *ipconfig* command line program and the Connection folder.	
description	A description for the adapter.	
mac_address	MAC address of the adapter.	
addresses	A pointer to the first **IP_ADDRESS** structure in a linked list of IP unicast addresses for the adapter.	
gateways	A pointer to the first **IP_ADDRESS** structure in a linked list of gateways for the adapter.	
dns	A pointer to the first **IP_ADDRESS** structure in a linked list of DNS server addresses for the adapter.	
ipv4_netmask	A pointer to the **IP_ADDRESS** structure of subnet mask for the adapter	

wifi_info	A pointer to the **WIFI_INFO** structure which stores the Wi-Fi information. It can be NULL if the interface is not **IF_TYPE_IEEE80211** or the Wi-Fi information is unavailable.
mobile_broadband_info	A pointer to the **MBN_INFO** structure which stores the mobile broadband network information. It can be NULL if the interface is not **IF_TYPE_WWANPP**, **IF_TYPE_WWANPP2 IF_TYPE_WWANPP**, or the information is unabailable.
reserved	Reserved for future use.

```
EApiStatus_t EApiGetAllNetworkInterfacesInfo (
      INOUT    PNETWORK_INFO *pOutput
    );
```

Description:

Get network information list.

Parameters:

pOutput

__INOUT Pointer to a list of NETWORK_INFO values that represent the network interface information supported by the device. The calling application must free the allocated memory by calling ***EApiDestroyNetworkInfoObject***.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get network information.
EAPI_STATUS_INVALID_PARAMETER	pOutput is NULL.
EAPI_STATUS_NOT_FOUND	There is no network interface.

```
EApiStatus_t EApiDestroyNetworkInfoObject (
      INOUT    PNETWORK_INFO pOutput
    );
```

Description:

Destroys an existing *NETWORK_INFO* object.

Parameters:

pOutput

__INOUT Pointer to the NETWORK_INFO object created by EApiGetAllNetworkInterfacesInfo.

Return Status Code:

EAPI_STATUS_SUCCESS	Success

```
EApiStatus_t EApiGetAllNetworkInterfacesJsonInfo (
      INOUT    char **pOutput
   );
```

Description:

Get network information list in JSON format string.

Parameters:

pOutput

__INOUT Pointer to JSON string buffer that represent the network interface information supported by the device. The calling application must free the allocated memory by calling ***EApiFreeJsonBuffer*** or ***free***.

Return Status Code:

EAPI_STATUS_SUCCESS	Succeed to get network information.
EAPI_STATUS_INVALID_PARAMETER	pOutput is NULL.
EAPI_STATUS_NOT_FOUND	There is no network interface.

Sample JSON Output:

```
[
    {
        "Index": 8,
        "Name": "Ethernet 2",
        "UUID": "{7B858C34-27D5-40A4-B977-1D60E43253FA}",
        "Description": "Realtek RTL8139/810x Family Fast Ethernet NIC",
        "MACAddress": "XX-XX-XX-XX-XX-XX",
        "Connected": true,
        "IsDefault": true,
        "DHCPEnabled": false,
        "OperStatus": 1,
        "Type": 6,
        "MTU": 1500,
        "Flags": 449,
        "IP": [
            {
                "Address": "fe80::1be:d2ed:6900:40e3",
                "IPv6": true
            },
            {
                "Address": "172.16.13.111",
                "IPv6": false
            }
            // ...
        ],
        "Gateway": [
            {
                "Address": "172.16.13.254",
                "IPv6": false
            }
        ],
        "DNS": [
            {
                "Address": "172.20.1.100",
                "IPv6": false
            }
            // ...
        ],
        "Netmask": "255.255.254.0"
    },
    {
        "Index": 10,
        "Name": "Wi-Fi",
        // ...
        "Profile": "Advantecher",
        "SSID": "Advantecher",
        "BssType": 1,
        "SignalStrength": -51,
        "SignalQuality": 99,
```

```
        "ReceivingRate": 300000,
        "TransmissionRate": 300000,
        "SecurityEnabled": true,
        "OneXEnabled": true,
        "AuthAlgorithm": "WPA",
        "CipherAlgorithm": "CCMP"
    },
    {
        "Index": 16,
        "Name": "Cellular 5",
        // ...
        "Manufacturer": "Sierra Wireless, Incorporated",
        "Model": "MC7304",
        "Firmware": "SWI9X15C 06.03.32.04 9904567 05",
        "CellularClass": "GSM",
        "BandClass": "3",
        "DataClasses": "GPRS, EDGE, UMTS, HSDPA, LTE, HSUPA, HSPA+",
        "IMEI": "490154203237518",
        "SubscriberID": "46697123456789", // IMSI or MIN
        "SimIccID": "89886920041XXXXXXXXX",
        "CurrentDataClass": "LTE",
        "SignalStrength": -53,
        "SignalQuality": 96,
        "PhoneNumber": [
            {
                "Number": "0000000000"
            }
        ],
        "APN": [
            {
                "AccessString": "INTERNET",
                "UserName": "",
                "Password": "",
                "Compression": false,
                "AuthProtocol": "NONE"
            }
        ]
    }
]
```


Key	Description
Common Fields	
Index	The interface index.
Name	A user-friendly name for the adapter. For example: "Local Area Connection 1." This name appears in contexts such as the *ipconfig* command line program and the Connection folder.
UUID	The unique ID for the adapter.
Description	A description for the adapter.
MACAddress	MAC address of the adapter.
Connected	Indicates whether adapter is connected to a network. (IfOperStatusUp)
IsDefault	Indicates whether adapter is a default route.
DHCPEnabled	Indicates whether DHCP is enabled on this adapter.
OperStatus	The operational status for the interface as defined in RFC 2863. For more information, see http://www.ietf.org/rfc/rfc2863.txt. This member can be one of the values from the IF_OPER_STATUS enumeration type defined in the Iftypes.h header file. On Windows Vista and later, the header files were reorganized and this enumeration is defined in the Ifdef.h header file.

Value	Meaning
IfOperStatusUp 1	The interface is up and able to pass packets.
IfOperStatusDown 2	The interface is down and not in a condition to pass packets. The **IfOperStatusDown** state has two meanings, depending on the value of **AdminStatus** member. If **AdminStatus** is not set to **NET_IF_ADMIN_STATUS_DOWN** and **ifOperStatus** is set to **IfOperStatusDown** then a fault condition is presumed to exist on the interface. If **AdminStatus** is set to **IfOperStatusDown**, then **ifOperStatus** will normally also be set to **IfOperStatusDown** or **IfOperStatusNotPresent** and there is not necessarily a fault condition on the interface.
IfOperStatusTesting 3	The interface is in testing mode.
IfOperStatusUnknown 4	The operational status of the interface is unknown.
IfOperStatusDormant 5	The interface is not actually in a condition to pass packets (it is not up), but is in a pending state, waiting for some external event. For on-demand interfaces, this new state identifies the situation where the interface is waiting for events to place it in the **IfOperStatusUp** state.
IfOperStatusNotPresent 6	A refinement on the **IfOperStatusDown** state which indicates that the relevant interface is down specifically because some component (typically, a hardware component) is not present in the

		managed system.
	IfOperStatusLowerLayerDown 7	A refinement on the **IfOperStatusDown** state. This new state indicates that this interface runs on top of one or more other interfaces and that this interface is down specifically because one or more of these lower-layer interfaces are down.

Type

The interface type as defined by the Internet Assigned Names Authority (IANA). Possible values for the interface type are listed in the Ipifcons.h header file. The table below lists common values for the interface type although many other values are possible.

Value	Meaning
IF_TYPE_OTHER 1	Some other type of network interface.
IF_TYPE_ETHERNET_CSMACD 6	An Ethernet network interface.
IF_TYPE_ISO88025_TOKENRING 9	A token ring network interface.
IF_TYPE_PPP 23	A PPP network interface.
IF_TYPE_SOFTWARE_LOOPBACK 24	A software loopback network interface.
IF_TYPE_ATM 37	An ATM network interface.
IF_TYPE_IEEE80211 71	An IEEE 802.11 wireless network interface. On Windows Vista and later, wireless network cards are reported as **IF_TYPE_IEEE80211**. On earlier versions of Windows, wireless network cards are reported as **IF_TYPE_ETHERNET_CSMACD**. On Windows XP with SP3 and on Windows XP with SP2 x86 with the

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>Wireless LAN API for Windows XP with SP2 installed, the WlanEnumInterfaces function can be used to enumerate wireless interfaces on the local computer.</td></tr>
<tr><td>IF_TYPE_TUNNEL
131</td><td>A tunnel type encapsulation network interface.</td></tr>
<tr><td>IF_TYPE_IEEE1394
144</td><td>An IEEE 1394 (Firewire) high performance serial bus network interface.</td></tr>
<tr><td>IF_TYPE_WWANPP
243</td><td>WWAN devices based on GSM technology.</td></tr>
<tr><td>IF_TYPE_WWANPP2
244</td><td>WWAN devices based on CDMA technology.</td></tr>
</table>
</td></tr>
<tr><td>MTU</td><td>The maximum transmission unit (MTU) size, in bytes.</td></tr>
<tr><td>Flags</td><td>A set of flags specifying various settings for the adapter.
<table>
<tr><td>IP_ADAPTER_DDNS_ENABLED</td><td>0x00000001</td></tr>
<tr><td>IP_ADAPTER_REGISTER_ADAPTER_SUFFIX</td><td>0x00000002</td></tr>
<tr><td>IP_ADAPTER_DHCP_ENABLED</td><td>0x00000004</td></tr>
<tr><td>IP_ADAPTER_RECEIVE_ONLY</td><td>0x00000008</td></tr>
<tr><td>IP_ADAPTER_NO_MULTICAST</td><td>0x00000010</td></tr>
<tr><td>IP_ADAPTER_IPV6_OTHER_STATEFUL_CONFIG</td><td>0x00000020</td></tr>
<tr><td>IP_ADAPTER_NETBIOS_OVER_TCPIP_ENABLED</td><td>0x00000040</td></tr>
<tr><td>IP_ADAPTER_IPV4_ENABLED</td><td>0x00000080</td></tr>
<tr><td>IP_ADAPTER_IPV6_ENABLED</td><td>0x00000100</td></tr>
<tr><td>IP_ADAPTER_IPV6_MANAGE_ADDRESS_CONFIG</td><td>0x00000200</td></tr>
</table>
</td></tr>
<tr><td>IP</td><td>IP unicast addresses for the adapter.
<table>
<tr><td>Address</td><td>The IP address</td></tr>
<tr><td>IPv6</td><td>Indicates whether IP is an IPv6 address.</td></tr>
</table>
</td></tr>
<tr><td>Gateway</td><td>Gateways for the adapter.
<table>
<tr><td>Address</td><td>The IP address</td></tr>
<tr><td>IPv6</td><td>Indicates whether IP is an IPv6 address.</td></tr>
</table>
</td></tr>
<tr><td>DNS</td><td>DNS server addresses for the adapter.
<table>
<tr><td>Address</td><td>The IP address</td></tr>
<tr><td>IPv6</td><td>Indicates whether IP is an IPv6 address.</td></tr>
</table>
</td></tr>
<tr><td>Netmask</td><td>Subnet mask for the adapter</td></tr>
</table>

Wi-Fi Fields	
Profile	The name of the profile used for the connection.
SSID	The SSID of the association.
BssType	Basic service set (BSS) network type. 1 -> infrastructure BSS network, 2 -> independent BSS (IBSS) network (ad hoc) 3 -> either infrastructure or IBSS network.
SignalStrength	Actual RSSI signal strength (dbm)
SignalQuality	A percentage value that represents the signal quality of the network.
ReceivingRate	The receiving rate of the association.
TransmissionRate	The transmission rate of the association.
SecurityEnabled	Indicates whether security is enabled for this connection.
OneXEnabled	Indicates whether 802.1X is enabled for this connection.
AuthAlgorithm	The authentication algorithm. It could be "802.11 Open", "802.11 Shared", "WPA", "WPA-PSK", "WPA-None", "RSNA", or "RSNA with PSK".
CipherAlgorithm	The cipher algorithm. It could be "None", "WEP-40", "TKIP", "CCMP", "WEP-104", or "WEP".
Mobile Broadband Fields	
Manufacturer	The name of the device manufacturer. It can be empty.
Model	The device model. It can be empty.
Firmware	The firmware-specific information for this device. It can be empty.
CellularClass	The type of cellular device. It could be "NONE", "GSM", "CDMA", or "*** UNKNOWN ***".
profile	The name of the profile used for the connection.
IMEI	IMEI (up to 15 digits) for GSM devices or ESN (11 digits) / MEID (17 digits) for CDMA devices.
SubscriberID	The subscriber ID of the device. For GSM device this represents the International Mobile Equipment Identity (IMSI) string (up to 15 digits). For CDMA device this represents the Mobile Identification Number (MIN) string or the International Roaming MIN (IRM) string (10 digits).
BandClass	The frequency band classes.
DataClasses	Specifies which data services are supported. For GSM devices, only the GSM-based data services can be present, that is, only GPRS, EDGE, UMTS, LTE, and HSDPA are valid values for GSM devices. For CDMA devices, only the CDMA-related data services will be present, that is, only 1xRTT, 1xEV-DO, and 1xEV-DO RevA are valid values for CDMA devices. 1xEV-DO RevB is reserved for future use.

<table>
<tr><td>SimIccID</td><td>The SIM International circuit card number (SimICCID) for the device.</td></tr>
<tr><td>CurrentDataClass</td><td>The current data class in the current network.</td></tr>
<tr><td>SignalStrength</td><td>Actual RSSI signal strength (dbm)</td></tr>
<tr><td>SignalQuality</td><td>A percentage value that represents the signal quality of the network.</td></tr>
<tr><td>PhoneNumber</td><td>The telephone numbers associated with the device.<table><tr><td>Number</td><td>Telephone number</td></tr></table></td></tr>
<tr><td>APN</td><td>Connection contexts.<table><tr><td>AccessString</td><td>Contains connection-specific access information. In GSM networks, this would be an access point name (APN) such as "data.thephone-company.com". In CDMA networks, this might be a special dial code such as "#777" or a NAI (Network Access Identifier) such as "somebody@thephone-company.com".</td></tr><tr><td>UserName</td><td>Contains the user name that is used for authentication.</td></tr><tr><td>Password</td><td>Contains the password that is used for authentication.</td></tr><tr><td>Compression</td><td>Specifies whether compression is to be used in the data link for header and data. This member is applicable only for GSM devices.</td></tr><tr><td>AuthProtocol</td><td>Indicates the type of compression used for PDP (Packet Data Protocol) activation. It could be "NONE", "PAP", "CHAP", "MsCHAPv2", or "*** UNKNOWN ***"</td></tr></table></td></tr>
</table>