
Title: SX UVL Client SDK for Win32

Drawing Type: API Specifications

Drawing No.: SC02790XH

Date: August 01, 2016

Rev	Description	Date
XX	First edition	Jul.13, 2005
XA	Second edition Modified clerical errors	Sep.8, 2005
XB	Third edition Added API	Jan.17, 2007
XC	Fourth edition Added API	Sep.30,2009
XD	Fifth edition Modified SXUSBDEVICE structure	Nov.12,2010
XE	Sixth edition Added support OS	Apr.12,2013i
XF	Seventh edition Added support OS Added API	Mar. 19,2014
XG	Eighth edition Added support OS	Jul. 21, 2015
XH	Nine edition Added API	Aug.01,2016

1. Overview.....	4
2. Specification.....	4
2.1. Operating Environment.....	4
3. Device server communication API specifications.....	5
3.1. Device server search process.....	5
3.2. USB device list obtaining process.....	6
3.3. Device connection process.....	8
3.4. Enhanced device connection process.....	8
3.5. Enhanced device connection process.....	9
3.6. Device disconnection process.....	10
3.7. Enhanced device disconnection process.....	10
3.8. Status receiving process.....	11
3.9. Device server information acquisition process.....	12
3.10. Device Server search rule number setup process.....	12
3.11. Device server reboot process.....	13
3.12. System status retrieval process.....	13
4. Disconnect request API.....	14
4.1. Initialization process for disconnect request API.....	14
4.2. Disconnect request API termination process.....	15
4.3. Receipt process for disconnect request.....	15
4.4. Approval process for disconnect process.....	16
4.5. Denial process for disconnect request.....	16
4.6. Completion notice of disconnect process.....	16
4.7. Disconnect request process.....	16
4.8. Transmission process of disconnect request.....	17
4.9. Cancel process for disconnect request.....	17
4.10. Handle closing process for disconnect request.....	18
4.11. SRU_NOTIFY message.....	19
4.12. Disconnect request message.....	20
5. Device driver API.....	21
5.1. Instance ID retrieval process.....	21
5.2. Instance ID release process.....	22

1. Overview

This document describes how to use the DLL developed to allow control of the basic functions of SX Virtual Link (excluding GUI) by a user application. SX Virtual Link is the utility used to manage silex device servers.

2. Specification

2.1. Operating Environment

Item	Description
OS	<p>Microsoft Windows 2000</p> <p>Microsoft Windows XP (32bit / 64bit)</p> <p>Microsoft Windows Vista (32bit / 64bit)</p> <p>Microsoft Windows 7 (32bit / 64bit)</p> <p>Microsoft Windows 8 (32bit / 64bit)*</p> <p>Microsoft Windows 8.1 (32bit / 64bit)*</p> <p>Microsoft Windows 10 (32bit / 64bit)*</p> <p>* Classic desktop applications are supported while Modern UI applications are not supported.</p> <p>* Windows RT is not supported.</p>
Compiler	<p>Microsoft Visual C++ 6.0 SP6</p> <p>Microsoft Visual Studio 2005</p> <p>Microsoft Visual Studio 2008</p> <p>Microsoft Visual Studio 2010</p> <p>Microsoft Visual Studio 2012</p> <p>Microsoft Visual Studio 2013</p>

3. Device server communication API specifications

3.1. Device server search process

SxuftpEnumDeviceServers

SxuftpEnumDeviceServers searches the specified broadcast address and lists the device servers on the network.

```

BOOL SxuftpEnumDeviceServers(
    LPDWORD lpdwBroadcasts, /* Array address of broadcast address*/
    DWORD dwCount,          /* Number of broadcast addresses */
    LPVOID lpbServers,      /* Array address of the structure */
    DWORD cbBuf,            /* Number of bytes of array */
    LPDWORD lpdwReaded,     /* Parameter's address to which the number of byte copied
                             will be returned*/
    LPDWORD lpdwReturned    /* Parameter's address to which the number of structure
                             copied will be returned*/
);

```

Parameter	Description
lpdwBroadcasts	Pointer for broadcast address array. Specify the broadcast address to search for the device server. Multiple broadcast addresses can be specified as array of DWORD value. When NULL is specified, device server in local address (255.255.255.255) will be searched.
dwCount	Specifies the number of broadcast addresses designated at lpdwBroadcasts. When NULL is set at lpdwBroadcasts, this parameter will be ignored.
lpbServers	Pointer indicating the SXPSEVER structure array. It receives the device server search result.
cbBuf	Specifies the size, by byte, of array pointed by lpbServers.
lpdwReaded	Indicates the number of byte copied to lpbServers.
lpdwReturned	Pointer indicating the parameter to which the number of device server searched is returned.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

Note

SXPSEVER structure

```

typedef struct _SXPSEVER {
    BYTE bNodeaddr[6];          /* Ethernet address */
    DWORD dwIp;                 /* IP address */
    char szMachineType[16];     /* Device server model name */
    char szHostName[16];       /* Host name */
} SXPSEVER, *LPSXPSEVER;

```

3.2. USB device list obtaining process

SxuptpEnumDevices

SxuptpEnumDevices lists the USB devices connected to the specified device server.

```

BOOL SxuptpEnumDevices (
    DWORD    dwIpaddr,        /* Device server IP address*/
    LPVOID    lpbDevices,     /* Array address of the structure */
    DWORD    cbBuf,           /* Number of bytes of array */
    LPDWORD   lpdwReaded,     /* Parameter address to which the number of byte
                              copied will be returned */
    LPDWORD   lpdwReturned    /* Parameter address to which the number of structure
                              copied will be returned */
);

```

Parameter	Description
dwIpaddr	Specify IP address of the device server of which the USB device will be listed.
lpbDevices	Pointer indicating the array of SXUSBDEVICE structure. It receives the USB device list.
cbBuf	Specifies the size, by byte, of array pointed by lpbDevices.
lpdwReaded	Indicates the number of byte copied to lpbDevices.
lpdwReturned	Pointer indicating the parameter to which the number of USB devices listed is returned.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

Note

SXUSBDEVICE structure

```
typedef struct _SXUSBDEVICE {
    char        szPortName[32];    /* USB logical port name */
    char        szDeviceName[64]; /* Device Name */
    DWORD       dwIpPc;           /* IP address of connected PC */
    WORD        wStatus;          /* Status information */
    WORD        wVid;             /* Device VID */
    WORD        wPid;             /* Device PID */
    char        szLocation[16];    /* USB port info which the device is connected to */
    WORD        wClass;           /* USB Interface Class */
} SXUSBDEVICE, *LPSXUSBDEVICE;
```

Parameter	Description																																
szPortName	USB logical port name of the server. This parameter is necessary for connection and disconnection.																																
szDeviceName	USB device name. Maximum length is 63 characters.																																
dwIpPc	IP address of the PC having sole possession of device. It is displayed as 0.0.0.0 when no device is connected over SVL.																																
wStatus	Device status is indicated in BIT as below. <table> <tr><td>0x8000</td><td></td></tr> <tr><td>0x4000</td><td>Disconnect OK/NG.</td></tr> <tr><td>0x2000</td><td>Encrypted communication on/off.</td></tr> <tr><td>0x1000</td><td>Device server supporting encrypted communication.</td></tr> <tr><td>0x0800</td><td>Wireless on/off.</td></tr> <tr><td>0x0400</td><td>USB devices supporting isochronous transmission</td></tr> <tr><td>0x0200</td><td>USB error occurred</td></tr> <tr><td>0x0100</td><td></td></tr> <tr><td>0x0080</td><td>Fixed to low.</td></tr> <tr><td>0x0040</td><td>Fixed to high.</td></tr> <tr><td>0x0020</td><td></td></tr> <tr><td>0x0010</td><td>Connect password on/off.</td></tr> <tr><td>0x0008</td><td>Device server supporting connect password.</td></tr> <tr><td>0x0004</td><td>(TBD)</td></tr> <tr><td>0x0002</td><td></td></tr> <tr><td>0x0001</td><td>Device searched/not.</td></tr> </table>	0x8000		0x4000	Disconnect OK/NG.	0x2000	Encrypted communication on/off.	0x1000	Device server supporting encrypted communication.	0x0800	Wireless on/off.	0x0400	USB devices supporting isochronous transmission	0x0200	USB error occurred	0x0100		0x0080	Fixed to low.	0x0040	Fixed to high.	0x0020		0x0010	Connect password on/off.	0x0008	Device server supporting connect password.	0x0004	(TBD)	0x0002		0x0001	Device searched/not.
0x8000																																	
0x4000	Disconnect OK/NG.																																
0x2000	Encrypted communication on/off.																																
0x1000	Device server supporting encrypted communication.																																
0x0800	Wireless on/off.																																
0x0400	USB devices supporting isochronous transmission																																
0x0200	USB error occurred																																
0x0100																																	
0x0080	Fixed to low.																																
0x0040	Fixed to high.																																
0x0020																																	
0x0010	Connect password on/off.																																
0x0008	Device server supporting connect password.																																
0x0004	(TBD)																																
0x0002																																	
0x0001	Device searched/not.																																
wVid	Indicates the VID of USB device.																																
wPID	Indicates the PID of USB device.																																
szLocation	Indicates which port the device is connected to (NULL terminated string). Character indicating the connection speed (H:High/F:Full/L:Low) is indicated in the first 1 byte, and it is followed by host controller device number, root hub port number, and hub port number. Each number uses the character from 0 to 9 and A to F. For example when connected to host controller device number 1 + root hub port number 1 + hub port number 1, this parameter is "H111".																																
wClass	Indicates the USB Interface Class of the device. *The values for USB Interface Class are as defined at USB.org.																																

3.3. Device connection process

SxuotpDeviceConnect

SxuotpDeviceConnect will connect the devices connected to the specified device server.

SxuotpDeviceConnect is the function for the old version of device server. Normally, please use SxuotpDeviceConnectEx.

```

BOOL SxuotpDeviceConnect(
    DWORD    dwIpaddr,        /* Device server IP address */
    LPBYTE    lpbNodeaddr,    /* Pointer to the variable that Ethernet address is stored
    */
    LPSTR      lpszPortName,   /* USB logical port name */
    BOOL       bFlags          /* Option */
);

```

Parameter	Description
dwIpaddr	Specifies device server IP address.
lpbNodeaddr	Specifies Ethernet address of the device server for which connection will be requested.
lpszPortName	Specifies USB logical port name for the devices to connect.
bFlags	Always specify 0.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

3.4. Enhanced device connection process

SxuotpDeviceConnectEx will connect the devices connected to the specified device server.

```

BOOL SxuotpDeviceConnectEx(
    DWORD    dwIpaddr,        /* Device server IP address */
    LPBYTE    lpbNodeaddr,    /* Pointer to the variable that Ethernet address is stored
    */
    LPSXUSBDEVICE lpDevice,    /* USB device information */
    BOOL       bFlags          /* Option */
);

```

Parameter	Description
dwIpaddr	Specifies device server IP address.
lpbNodeaddr	Specifies Ethernet address of the device server for which connection will be requested.
lpDevice	Specifies the pointer that points to the SWUSBDEVICE structure of the USB device to be connected.
bFlags	Always specify 0.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

3.5. Enhanced device connection process

SxuptpDeviceConnectEx2

SxuptpDeviceConnectEx2 will connect the devices connected to the specified device server.

```

BOOL SxuptpDeviceConnectEx2(
    DWORD      dwIpaddr,      /* Device server IP address */
    LPBYTE      lpbNodeaddr,  /* Pointer to the variable that Ethernet address is stored */
    LPSXUSBDEVICE lpDevice,   /* USB device information */
    DWORD      dwFlags,       /* Option flags */
    LPVOID      lpbOption     /* Option information */
);

```

Parameter	Description	
dwIpaddr	Specifies device server IP address.	
lpbNodeaddr	Specifies Ethernet address of the device server for which connection will be requested.	
lpDevice	Specifies the pointer that points to the SWUSBDEVICE structure of the USB device to be connected.	
dwFlags	Value	Content
	SVLC_ENCRYPT	It encrypts data communication with the USB device.
	SVLC_AUTH	Specifies the connect password. If you connect the USB device connection password is enabled, you must specify this flags. If you use this flag, please specify the pointer of NULL terminated string that contains the password string (CHAR type) to lpbOption.
lpbOption	Specifies the additional information in accordance with the value of dwFlags.	

Return value

When the function is successfully completed, status below will be returned:

Value	Description
0	USB device is connectable.
1	Network communication error.
2	Encryption error.

3.6. Device disconnection process

SxuotpDeviceDisconnect

SxuotpDeviceDisconnect will disconnect the devices connected to the specified device server. SxuotpDeviceDisconnect is the function for the old version of device server. Normally, please use SxuotpDeviceDisconnectEx.

```

BOOL SxuotpDeviceDisconnect(
    DWORD    dwIpaddr,        /* Device server IP address */
    LPBYTE    lpbNodeaddr,    /* Pointer to the variable that Ethernet address is stored
    */
    LPSTR     lpszPortName    /* USB logical port name */
);

```

Parameter	Description
dwIpaddr	Specifies device server IP address.
lpbNodeaddr	Specifies Ethernet address of the device server for which disconnection will be requested.
lpszPortName	Specifies USB logical port name for the devices to disconnect.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

3.7. Enhanced device disconnection process

SxuotpDeviceDisconnectEx

SxuotpDeviceDisconnectEx will disconnect the devices connected to the specified device server.

```

BOOL SxuotpDeviceDisconnectEx(
    DWORD    dwIpaddr,        /* Device server IP address */
    LPBYTE    lpbNodeaddr,    /* Pointer to the variable that Ethernet address is stored
    */
    LPSXUSBDEVICE lpDevice    /* USB device information */
);

```

Parameter	Description
dwIpaddr	Specifies device server IP address.
lpbNodeaddr	Specifies Ethernet address of the device server for which disconnection will be requested.
lpDevice	Specifies the pointer that points to the SWUSBDEVICE structure of the USB device to be disconnected.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

3.8. Status receiving process

SxuptpGetDeviceStatus

SxuptpGetDeviceStatus will acquire status information of USB devices connected to the specified device server.

```
DWORD SxuptpGetDeviceStatus(  
    DWORD    dwIpaddr,        /* Device server IP address */  
    LPSTR     lpszPortName    /* Specifies USB logical port name for a device */  
);
```

Parameter	Description
dwIpaddr	Specifies device server IP address.
lpszPortName	Specifies USB logical port name for a device

Return value

When the function is successfully completed, status below will be returned:

Value	Description
0x00000001	USB device is connectable.
0x00000002	USB device is connected to a computer.
0x00000003	USB device is being accessed by other user.
0x00000004	USB device is not detected.
0xFFFFFFFF	Network communication error

3.9. Device server information acquisition process

SxuptpGetDeviceServerInfo

SxuptpGetDeviceServerInfo will acquire information about device server whose IP address is specified.

```

BOOL SxuptpGetDeviceServerInfo (
    DWORD    dwIpaddr,        /* Device server IP address */
    LPVOID    lpbServers,     /* Array address of the structure */
    DWORD    cbBuf            /* Number of bytes of array */
);

```

Parameter	Description
dwIpaddr	Specifies device server IP address.
lpbServers	Pointer indicating the SXPSEVER structure array. It receives the device server information.
cbBuf	Specifies the size, by byte, of array pointed by lpbServers.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

Note

SXPSEVER structure

```

typedef struct _SXPSEVER {
    BYTE    bNodeaddr[6];      /* Ethernet address */
    DWORD    dwIp;             /* IP address */
    char     szMachineType[16]; /* Device server model name */
    char     szHostName[16];    /* Host name */
} SXPSEVER, *LPSXPSEVER;

```

3.10. Device Server search rule number setup process

SxuptpSetSrchrRuleCookie

SxuptpSetSrchrRuleCookie will set the vender the unique check number that is used during device server search. Only the device server with the same rule number can be searched with SxuptpEnumDeviceServers.

```

BOOL SxuptpSetSrchrRuleCookie(
    WORD    wRuleCookie        /* Rule number */
);

```

Parameter	Description
wRuleCookie	Specifies the rule number.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

3.11. Device server reboot process

SxuptpRebootDeviceServer

SxuptpRebootDeviceServer performs a remote reboot of the device server.

```

BOOL SxuptpRebootDeviceServer(
    DWORD    dwIpaddr,        /* Device server IP address */
    LPBYTE    lpbNodeaddr     /* Pointer to variable where Ethernet address is stored */
);

```

Parameter	Meaning
dwIpaddr	Specifies the IP address of the device server.
lpbNodeaddr	Specifies the Ethernet address of the device server for which a reboot will be requested.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

3.12. System status retrieval process

SxuptpGetSystemStatus

SxuptpGetSystemStatus retrieves system status information of the specified device servers.

```

BOOL SxuptpGetSystemStatus(
    DWORD    dwIpaddr,        /* Device server IP address */
    LPSTR     lpszStatus,     /* Pointer to variable where system status is stored */
    DWORD    cbBuf            /* Number of bytes of array */
);

```

Parameter	Meaning
dwIpaddr	Specifies the IP address of the device server.
lpszStatus	Specifies the pointer of array to store a system status string retrieved from the device server.
cbBuf	Specifies size of array pointed by lpszStatus in byte.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

System status may not be supported depending on model of device servers. In such a case, the function will fail. Also, system status information will differ on each device server model.

4. Disconnect request API

To use disconnect request API, call SruInitialize function once within the program. Also, when completed using the disconnect request API, call SruUninitialize function.

As SruInitialize function is called, the disconnect request becomes ready. The disconnect request event will be notified to a program by Windows message. Once the message is received, the program can execute each function of disconnect request by calling APIs appropriate for the message.

The disconnect request API cannot be called from a worker thread. Please be sure to call it from a main thread.

4.1. Initialization process for disconnect request API

SruInitialize

SruInitialize performs necessary initialization process to use the disconnect request API.

The applications will need to call SruInitialize function in order to use the disconnect request API.

```

BOOL SruInitialize(
    HINSTANCE hInst,          /* Instance handle */
    HWND hParent,            /* Window handle to notify disconnect request event */
    LPCSTR lpszMyName        /* Name to use in communication */
);

```

Parameter	Meaning
hInst	Specifies the instance handle of application which uses the disconnect request API.
hParent	Specifies the handle of window to notify Windows message when disconnect request event occurs.
lpszMyName	Specifies the name string to notify to receiver of the disconnect request when it is sent or replied. For lpszMyName, up to 64 byte (including NULL) can be specified.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.2. Disconnect request API termination process

SruUninitialize

SruUninitialize performs termination process for disconnect request API. The application needs to call the SruUninitialize function to terminate use of the disconnect request API.

```
BOOL SruUninitialize(
    HINSTANCE hInst          /* Instance handle */
);
```

Parameter	Meaning
hInst	Specifies the instance handle of application which has been specified when SruInitialize function was called.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.3. Receipt process for disconnect request

SruSendReply

SruSendReply notifies receipt of the disconnect request to the sender.

```
BOOL SruSendReply(
    WORD      wAutoDisconnect,    /* Auto disconnect approval flag */
    DWORD     dwTimeout          /* Auto disconnect timeout value */
);
```

Parameter	Meaning
wAutoDisconnect	Specifies whether to allow auto disconnect. Set 1 to allow and 0 to deny it.
dwTimeout	Specifies the timeout period until auto disconnect (sec). After amount of time (in sec) specified with dwTimeout has passed when wAutoDisconnect value is set to 1, the program needs to disconnect the target USB device upon receipt of the disconnect request.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

Explanation

Once the disconnect request receipt message (**SRU_REQUEST**) is received, application will be required to check if it is currently connecting the target USB device. If the USB device is connected, the application will need to make a response immediately using SruSendReply.

4.4. Approval process for disconnect process

SruAcceptRequest

SruAcceptRequest is called by application to give approval for disconnect request when it is received.

BOOL SruAcceptRequest(VOID);

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

Explanation

In order to give approval for disconnect request, applications will need to perform disconnect process for the target USB device after calling the SruAcceptRequest function. Also, after disconnect process is completed, applications will need to notify completion of disconnect process by calling the SruDeviceDisconnected function.

4.5. Denial process for disconnect request

SruRefuseRequest

SruRefuseRequest is called by application in order to deny disconnect request when disconnect request is received.

BOOL SruRefuseRequest(VOID);

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.6. Completion notice of disconnect process

SruDisconnectedDevice

After disconnect request is approved, SruDisconnectedDevice notifies completion of the USB device disconnection to the application which has sent the disconnect request.

BOOL SruDisconnectedDevice(VOID);

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.7. Disconnect request process

SruDeviceNotFound

When the USB device status is not detected or disconnect process fails after disconnect request is approved, SruDeviceNotFound notifies it to the application which has sent disconnect request.

BOOL SruDeviceNotFound(VOID);

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.8. Transmission process of disconnect request

SruSendRequest

SruSendRequest sends the disconnect request to the user who is currently connecting to the USB device.

```
HANDLE SruSendRequest(
    HWND          hParent,          /* Window handle to notify disconnect request event */
    DWORD          dwIp,            /* Device server IP address */
    LPBYTE         lpbNodeaddr,     /* Device server Ethernet address */
    LPSXUSBDEVICE lpDevice,         /* USB device information */
    DWORD          dwTimeout        /* Timeout value */
);
```

Parameter	Meaning
hParent	Specifies the handle of window to notify the disconnect request event. Event is notified using Windows message.
dwIp	Specifies the IP Address of device server which USB device of disconnect target is connected to.
lpbNodeaddr	Specifies the Ethernet Address of device server which USB device of disconnect target is connected to.
lpDevice	Specifies the pointer indicating SXUSBDEVICE structure for USB device of the disconnect target.
dwTimeout	Specifies the timeout for disconnect request. Timeout value is specified in sec. When set to 0, there will be no timeout. If communication is not processed successfully within the specified time period, the disconnect request will fail as communication error. The timeout value can be specified in 0-60 sec.

Return value

Handle indicating the disconnect request object is returned when the function is successfully completed, or NULL is returned when the function fails.

4.9. Cancel process for disconnect request

SruCancelRequest

SruCancelRequest cancels the disconnect request sent by SruSendRequest.

```
BOOL SruCancelRequest(
    HANDLE hRequest /* Handle of disconnect request object */
);
```

Parameter	Meaning
hRequest	Specifies the handle of disconnect request object returned by SruSendRequest function.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.10. Handle closing process for disconnect request

SruCloseHandle

SruCloseHandle closes handle of disconnect request object obtained by SruSendRequest. To finish sending the disconnect request, close the disconnect request object using the SruCloseHandle function.

```
BOOL SruCloseHandle(
    HANDLE    hRequest    /* Handle of disconnect request object */
);
```

Parameter	Meaning
hRequest	Specifies the handle of disconnect request object returned by SruSendRequest function.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.

4.11. SRU_NOTIFY message

SRU_NOTIFY is the message to notify occurrence of disconnect request events.

Parameter	Meaning
wParam	This is the pointer of SRUINFO structure which receives disconnect request information.
lParam	IP Address of computer to send disconnect request to

Explanation

SRU_NOTIFY is Windows message to notify occurrence of disconnect request events to the program. The message is notified to handle of window specified by SruInitialize or SruSendRequest functions. The disconnect request message type is stored in wMessage member of SRUINFO structure. The program needs to call the disconnect request API according to wMessage member type.

SRUINFO structure definition

```
typedef struct _SRUINFO {
    DWORD    dwSize;                /* SRUINFO structure size */
    WORD     wMessage;              /* Disconnect request message */
    char     szMyName[64];          /* Name of request sender */
    char     szDestName[64];        /* Name of request receiver */
    DWORD    dwSrvIp;               /* Device server IP address */
    BYTE     bNodeaddr[6];          /* Device server Ethernet address */
    WORD     wVid;                  /* Device VID */
    WORD     wPid;                  /* Device PID */
    char     szDeviceName[64];      /* Device name */
    char     szLocation[16];        /* Locational info of device connection */
    DWORD    dwRemainingTime;       /* Remaining time until auto disconnection(sec) */
} SRUINFO, *LPSRUINFO;
```

Parameter	Meaning
dwSize	SRUINFO structure size
wMessage	Stores message type of disconnect request. For the message type, refer to 4.12 Disconnect request message.
szMyName	Stores a name specified by the SruInitialize function.
szDestName	Stores a name of disconnect request receiver.
dwSrvIp	IP Address of device server
bNodeaddr	Ethernet Address of device server
wVid	Stores VID of USB device
wPid	Stores PID of USB device
szDeviceName	USB device name
szLocation	Indicates locational info of the connected device. In case of multi-functional devices, only information of first end point is stored.
dwRemainingTime	Stores the remaining time until auto disconnect when disconnect request is sent while the auto disconnect function is enabled (in sec).

4.12. Disconnect request message

Message	Meaning
SRU_REQUEST	This is notified when the program receives disconnect request. The program needs to check if the requested USB device is connected to itself, and reply a return value. If the program is connected to USB device, call the SruSendReply function. Also, return TRUE for SRU_NOTIFY message or else return FALSE.
SRU_CANCEL	This is notified when the disconnect request is cancelled.
SRU_ACCEPT	This is notified when the disconnect request is allowed.
SRU_REFUSE	This is notified when the disconnect request is denied.
SRU_DISCONNECT	This is notified when the USB device disconnection is completed. The program becomes ready to connect to the USB device when receiving this message after sending the disconnect request.
SRU_BUSY	This is notified when failed to receive the disconnect request due to communication being processed for another disconnect request.
SRU_REPLY	This is notified when there is a reply to disconnect request.
SRU_NOTFOUND	This is notified if the program (the one to receive the disconnect request) fails to detect the target USB device or disconnect from it.
SRU_REPLYEX	This is notified when there is a reply to disconnect request. This message indicates that there is a reply while auto disconnect is enabled.
SRU_DISCONTIME	This notifies the remaining time until auto disconnection. The program (the one to receive the disconnect request) will need to disconnect the USB device when dwRemainingTime value became 0. Call the API in below order: SruAcceptRequest SxuotpDisconnectDeviceEx SruDeviceDisconnected
SRU_CONFIRM	This is notified when the program needs to check if the USB device of disconnect request is connected to itself. After receiving this message, the program needs to check immediately if the requested USB device is connected to itself and then reply a return value. When USB device is connected, return TRUE for SRU_NOTIFY message or else return FALSE.
SRU_TIMEOUT	This is notified when there is no reply even after timeout period being specified by SruSendRequest function has passed.

5. Device driver API

5.1. Instance ID retrieval process

SxuptyGetInstanceId

SxuptyGetInstanceId retrieves the instance ID of USB device driver which is recognized via SXUPTP. When the target USB device is not connected, the function will fail.

* This function does not support Windows 2000.

```
DWORD SxuptyGetInstanceId(
    DWORD          dwIp,          /* Device server IP Address */
    LPCSTR         lpszLocation,  /* USB device location */
    LPSXDEVINST    *lpDevInst    /* Instance ID info */
);
```

Parameter	Meaning
dwIp	Specifies the IP Address of device server being connected to USB device to retrieve the instance ID.
lpszLocation	Specifies the locational information of the USB device to retrieve the instance ID.
*lpDevInst	Specifies the pointer to SXDEVINST structure pointer. The instance ID of USB device driver is saved to *lpDevInst. After calling this function, the application will need to release memory by calling SxuptyFreeInstanceId.

Return value

Number of retrieved instance ID is returned when the function is successfully completed. Usually, only 1 instance ID is returned when a USB device is connected, while as many instance IDs as device functionalities are returned when multi-functional device is connected.

Note

```
SXDEVINST structure
typedef struct _SXDEVINST {
    LPSTR      lpszInstanceId; /* Instance ID */
} SXDEVINST, *LPSXDEVINST;
```

5.2. Instance ID release process

SxuptpFreeInstanceId

SxuptpFreeInstanceId releases memory of SXDEVINST structure retrieved by SxuptpGetInstanceId function.

* This function does not support Windows 2000.

```
BOOL SxuptpFreeInstanceId(
    LPSXDEVINST lpDevInst    /* Instance ID info */
);
```

Parameter	Meaning
*lpDevInst	Specifies the pointer of SXDEVINST structure retrieved by SxuptpGetInstanceId function.

Return value

TRUE is returned when the function is successfully completed or else FALSE is returned.