

Sentinel SuperPro Borland C/C++ Interface

This document contains information on using the Sentinel SuperPro™ Borland C/C++ interface. It describes the interface requirements, build example information, API functions and the error codes.

Interface Requirements

This section contains information on what is required to use this interface.

Compiler Compatibility

This interface is compatible with the following compilers:

- Borland C++Builder 5.0
- Borland C++Builder 6.0

Specific Requirements

Here are the additional requirements for this interface:

- Sentinel System Driver 7.1.0 (or higher)
- Sentinel Protection Server 7.1.0 (or higher)

Platforms Support

This interface is supported on the following platforms:

- Windows 98
- Windows ME
- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003

Build Example Information

The following information can be used for building the example program given with this interface:

Evaluation Program Files

File Name	Description
<i>Activate.cpp</i>	The Activate form source code for the example program.
<i>Activate.dfm</i>	A binary file for the activate form.
<i>Activate.h</i>	The header file for the activate.cpp.
<i>Enumserver.cpp</i>	The EnumServer form source code for the example program.
<i>Enumserver.dfm</i>	A binary file for the EnumServer form.
<i>Enumserver.h</i>	The header file for the <i>EnumServer.cpp</i> .
<i>Eval.cpp</i>	The main form source code for the example program.
<i>Eval.dfm</i>	A binary file for the main form.
<i>Eval.h</i>	The header file for <i>eval.cpp</i> .
<i>Getkeyinfo.cpp</i>	The GetKeyInfo form source code for the example program.
<i>Getkeyinfo.dfm</i>	A binary file for the GetKeyInfo form.
<i>Getkeyinfo.h</i>	The header file for <i>GetKeyInfo.cpp</i> .
<i>Password.cpp</i>	The Passwords form source code for the example program.
<i>Password.dfm</i>	A binary file for the password form.
<i>Password.h</i>	The header file for <i>password.cpp</i> .
<i>SafeNet.ico</i>	The icon file for <i>sproeval.exe</i> .
<i>SPROEVAL.bpr</i>	A Borland Builder project file for the example program.
<i>SPROEVAL.cpp</i>	An entry point of the program.
<i>SPROEVAL.mak</i>	A project make file for the example program.
<i>SPROEVAL.res</i>	A resource file for the example program.
<i>Spromeeps.h</i>	An include file for the Sentinel SuperPro APIs.
<i>Spromeeps.lib</i>	The Sentinel SuperPro static link library .
<i>SuperPro Borland C++Builder Interface.pdf</i>	This document
<i>Sx32w.dll</i>	The Sentinel SuperPro dynamic link library.
<i>Sx32w.lib</i>	An import library for <i>sx32w.dll</i> .

Build Example Instructions

To build the sproeval example program, follow the steps given below:

1. Open *sproeval.bpr* in the Borland C++Builder.
2. From the **Build** menu, click **Build sproeval.exe** to build the example program.

Tip: In your application you need to use the *spromeps.h*, *spromeps.lib* or *sx32w.dll* (and *sx32w.lib*) to call the Sentinel SuperPro API functions.

Sentinel SuperPro API Functions

The RB_SPRO_APIPACKET Structure

The Sentinel driver uses the data in the RB_SPRO_APIPACKET structure to communicate with the key. You should always allocate memory for the structure and NEVER modify the data in it.

Packet Definition

```
typedef RB_DWORD[SPRO_APIPACKET_SIZE/sizeof(RB_DWORD)] RB_SPRO_APIPACKET;  
typedef RBP_VOID RBP_SPRO_APIPACKET;
```

The following APIs are supported by this interface:

Note: The subsequent pages discuss the various Sentinel SuperPro API functions. You should refer to the *Sentinel SuperPro Developer's Guide* to understand the Sentinel SuperPro concepts. You may also refer to the section on "Data Type, Constants and Structure Definitions" on page 33 for more details.

RNBOsproFormatPacket

This function initializes and validates the API packet based on its size.

Format

```
SP_STATUS SP_API RNBOsproFormatPacket(RBP_SPRO_APIPACKET packet,  
                                       RB_WORD packetSize);
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	OUT	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4. You should always allocate memory for the structure and NEVER modify the data in it.
<i>packetSize</i>	IN	RB_WORD	The size of the RB_SPRO_APIPACKET structure.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

You must call this function once before calling any other SuperPro API function.

RNBOSproInitialize

This function initializes the API packet and sets the values specified (if any) in the configuration file or the NSP_HOST environment variable.

Format

```
SP_STATUS SP_API RNBOSproInitialize (  
    RBP_SPRO_APIPACKET          thePacket );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	OUT	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4. You should always allocate memory for the structure and NEVER modify the data in it.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

You must call this function immediately after RNBOSproFormatPacket.

RNBOsproSetProtocol

This function sets the network protocol for allowing communication between the client and Sentinel Protection Server. You can choose from the following protocols: NetBEUI, TCP/IP, IPX and SAP. By default, TCP/IP is used.

Format

```
SP_STATUS SP_API RNBOsproSetProtocol (
    RBP_SPRO_APIPACKET          thePacket,
    PROTOCOL_FLAG                protocol );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>protocol</i>	IN	PROTOCOL_FLAG	<p>The protocol chosen by a client for communication with the Sentinel Protection Server. The valid values are:</p> <ul style="list-style-type: none"> <input type="checkbox"/> NSPRO_TCP_PROTOCOL = 1 <input type="checkbox"/> NSPRO_IPX_PROTOCOL = 2 <input type="checkbox"/> NSPRO_NETBEUI_PROTOCOL = 4 <input type="checkbox"/> NSPRO_SAP_PROTOCOL = 8†

†SAP protocol can also be used with direct IPX address, Preferred with Broadcast for speed.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- This function can be called after successfully calling RNBOsproInitialize and before calling RNBOsproFindFirstUnit.
- This function will not work if the packet already has a license and will return the SP_INVALID_OPERATION error code.
- Alternatively, your customers can set the network protocol in the configuration file you ship to them. However, a protocol set using the RNBOsproSetProtocol function will always override the value specified in the configuration file.

RNBOsproSetContactServer

This function sets the access mode for finding the key.

You may also set this function to the IP or IPX address, NetBEUI name or name of the system where the Sentinel Protection Server is running.

Format

```
SP_STATUS SP_API RNBOsproSetContactServer (  
    RBP_SPRO_APIPACKET                thePacket ,  
    RBP_CHAR                           serverName );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>serverName</i>	IN	RBP_CHAR	A pointer to the location that contains one of the following values: <ul style="list-style-type: none"><input type="checkbox"/> RNBO_STANDALONE<input type="checkbox"/> RNBO_SPN_DRIVER<input type="checkbox"/> RNBO_SPN_LOCAL<input type="checkbox"/> RNBO_SPN_BROADCAST<input type="checkbox"/> RNBO_SPN_ALL_MODES<input type="checkbox"/> RNBO_SPN_SERVER_MODES<input type="checkbox"/> IP address, IPX address, NetBEUI name or the workstation name. However, the name length cannot exceed 63 single-byte characters.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- You can call this function before calling RNBOsproFindFirstUnit. This function will not work if the packet already has a license and will return the SP_INVALID_OPERATION error code.
- An access mode can be set using three alternate methods: by calling the RNBOsproSetContactServer function, or by setting a tag in the configuration file, or by setting the NSP_HOST environment variable. RNBO_SPN_ALL_MODES will be used if none of the three methods is used.

An access mode set using RNBOsproSetContactServer has priority over the value set via the other two methods.

RNBOsproSetSharedLicense

This function allows you to enable/disable the main and sublicense sharing. The licenses issued to users from the same seat (a user name and MAC address combination) are shared.

Format

```
SP_STATUS SP_API RNBOsproSetSharedLicense(
    RBP_SPRO_APIPACKET thePacket,
    RB_WORD shareMainLic,
    RB_WORD shareSubLic);
```

Parameters

Name	Direction	Parameter Type	Description
<i>packet</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>shareMainLic</i>	IN	RB_WORD	Enables/disables the main license sharing. Use any of the following constants: <ul style="list-style-type: none"> <input type="checkbox"/> SP_ENABLE_MAINLIC_SHARING (enables main license sharing) <input type="checkbox"/> SP_DISABLE_MAINLIC_SHARING (disables main license sharing) By default, main license sharing is enabled.
<i>shareSubLic</i>	IN	RB_WORD	Enables/disables the sublicense sharing. Use any of the following constants: <ul style="list-style-type: none"> <input type="checkbox"/> SP_ENABLE_SUBLIC_SHARING (enables sublicense sharing) <input type="checkbox"/> SP_DISABLE_SUBLIC_SHARING (disables sublicense sharing) By default, sublicense sharing is disabled.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

You can call this function before calling RNBOsproFindFirstUnit.

RNBOsproFindFirstUnit

This function finds the first SuperPro key with the specified developer ID and obtains a license, if available.

Format

```
SP_STATUS SP_API RNBOsproFindFirstUnit (
    RBP_SPRO_APIPACKET          thePacket ,
    RB_WORD                      devleoperID );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>developerID</i>	IN	RB_WORD	The developer ID of the Sentinel SuperPro key to find.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

If RNBOsproFindFirstUnit is called with an API packet that already has a license, then the SP_INVALID_OPERATION error is returned.

RNBOsproGetContactServer

This function returns the access mode set to obtain a license.

Format

```
SP_STATUS SP_API RNBOsproGetContactServer (
    RBP_SPRO_APIPACKET          thePacket,
    RBP_CHAR                     serverNameBuf,
    RB_WORD                     serverNameBufSz );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	SPP_UPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>serverNameBuf</i>	OUT	RBP_CHAR	A pointer to the buffer in which the Sentinel Protection Server name is copied. Memory needs to be allocated for the buffer.
<i>serverNameBufSz</i>	IN	RB_WORD	The length of the buffer. The maximum length recommended is 64 bytes.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- You can call this function anytime after successfully calling RNBOsproFindFirstUnit.
- You could call this function in order to know from where the license was issued to your application. For example, you can display the Sentinel Protection Server name via some command in your user interface so that the user is aware of which system was contacted to obtain a license.

RNBOSproSetHeartBeat

This function sets the heartbeat interval for maintaining the communication between a client and the Sentinel Protection Server. The heartbeat time can be set to INFINITE_HEARTBEAT or from 1 minute to 30 days, in multiples of 1 second.

The heartbeat represents the interval within which your application notifies the Sentinel Protection Server that it is still running. If this function is not called, the protection server assumes the default value as two minutes (120 seconds). As a result, if no call is made to the key at least every two minutes, the license will be released and the SP_INVALID_LICENSE error will be returned if any call is made using the same packet.

Format

```
SP_STATUS SP_API RNBOSproSetHeartBeat (
    RBP_SPRO_APIPACKET          thePacket ,
    RB_DWORD                    heartBeatValue );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>heartBeatValue</i>	IN	RB_DWORD	A value that represents time in seconds. The valid values are: <input type="checkbox"/> LIC_UPDATE_INT = 120 <input type="checkbox"/> MAX_HEARTBEAT = 2592000 <input type="checkbox"/> MIN_HEARTBEAT = 60 <input type="checkbox"/> INFINITE_HEARTBEAT = -1

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- You should call this function after calling RNBOSproFindFirstUnit.
- Alternatively, your customers can set the heartbeat interval in the configuration file you shipped them. However, the time specified in RNBOSproSetHeartBeat will always override the value specified in the configuration file.
- When the heartbeat time interval set is too low and there is congestion on the network, then the application can terminate even before contacting the protection server. To avoid such a situation, you may want to specify a longer heartbeat interval. However, if infinite heartbeat set, then the protection server will never release the license unless RNBOSproReleaseLicense is called.

RNBOSproFindNextUnit

This API finds the next SuperPro key based on the developer ID maintained in the RB_SPRO_APIPACKET structure.

Format

```
SP_STATUS SP_API RNBOSproFindNextUnit (
    RBP_SPRO_APIPACKET thePacket );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- This function should be called when RNBOSproFindFirstUnit has returned the SP_NO_LICENSE_AVAILABLE error.
- If RNBOSproFindNextUnit returns success, the application will release the license obtained by the RNBOSproFindFirstUnit API call and will contain the data for the next SuperPro key. However, if the function returns an error, the RB_SPRO_APIPACKET structure will be marked invalid. To re-initialize the structure, use RNBOSproFindFirstUnit and optionally, RNBOSproFindNextUnit depending on the number of SuperPro keys found.

RNBOSproRead

This function reads a word at the specified address. If successful, the data variable will contain the word value.

Format

```
SP_STATUS SP_API RNBOSproRead (
    RBP_SPRO_APIPACKET          thePacket ,
    RB_WORD                     address ,
    RBP_WORD                     data );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>address</i>	IN	RB_WORD	The cell address to be read.
<i>data</i>	OUT	RBP_WORD	A pointer to the variable that will contain the data read from the key.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

If an attempt is made to read a non-readable word or algorithm/hidden word, the SP_ACCESS_DENIED error will be returned. For security reasons, algorithm words cannot be read.

RNBOsproExtendedRead

This function reads the word and access code at the specified address. On success, the data variable contains the information from the SuperPro key and the access code variable contains the access code.

Format

```
SP_STATUS SP_API RNBOsproExtendedRead (
    RBP_SPRO_APIPACKET          thePacket ,
    RB_WORD                     address ,
    RBP_WORD                     data ,
    RBP_BYTE                     accessCode );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>address</i>	IN	RB_WORD	The address to be read.
<i>data</i>	OUT	RBP_WORD	A pointer to the variable that will contain the data read from the key.
<i>accessCode</i>	OUT	RBP_BYTE	A pointer to the variable that will contain the access code associated with the word that was read.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

If an attempt is made to read a non-readable word or algorithm/hidden word, the SP_ACCESS_DENIED error will be returned. For security reasons, algorithm words cannot be read.

RNBOSproWrite

This function is used to write a word and its associated access code at the specified address. The word data is placed in the data variable and its associated access code in the access code variable.

Format

```
SP_STATUS SP_API RNBOSproWrite (
    RBP_SPRO_APIPACKET          thePacket,
    RB_WORD                     writePassword,
    RB_WORD                     address,
    RB_WORD                     data,
    RB_BYTE                     accessCode );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>writePassword</i>	IN	RB_WORD	The write password for the SuperPro key.
<i>address</i>	IN	RB_WORD	Contains the address of the cell where write is to be performed.
<i>data</i>	IN	RB_WORD	Contains the word to write in the key.
<i>accessCode</i>	IN	RB_BYTE	Contains the access code associated with the word to write.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- On success, the data and its associated access code are written at the specified address.
- If wrong write password is specified, the SP_ACCESS_DENIED error is returned.
- This function can be used to overwrite any word with read-write access on the SuperPro key (with the exception of the reserved words).

RNBOsproOverwrite

This function is used to change the value and access code of a word at the specified address. The word data is placed in the data variable and its associated access code in the access code variable.

Format

```
SP_STATUS SP_API RNBOsproOverwrite (
    RBP_SPRO_APIPACKET    thePacket,
    RB_WORD                writePassword,
    RB_WORD                overwritePassword1,
    RB_WORD                overwritePassword2,
    RB_WORD                address,
    RB_WORD                data,
    RB_BYTE                accessCode );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>writePassword</i>	IN	RB_WORD	The write password for the SuperPro key.
<i>overwritePassword1</i>	IN	RB_WORD	The overwrite password 1 for the SuperPro key.
<i>overwritePassword2</i>	IN	RB_WORD	The overwrite password 2 for the SuperPro key.
<i>address</i>	IN	RB_WORD	The address of the word to write.
<i>data</i>	IN	RB_WORD	Contains the word to write in the key.
<i>accessCode</i>	IN	RB_BYTE	Contains the access code associated with the word to write.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- You can use this function to overwrite words with access code 0. To overwrite words with other access codes, use the RNBOsproOverwrite function.
- If the write password was incorrect, the SP_ACCESS_DENIED error will be returned.
- This function can be used to overwrite any word on the SuperPro key (with the exception of the reserved words).

RNBOSproDecrement

This function is used to decrement a counter word by one.

Note: If the counter is associated with an active algorithm, and the counter is decremented to 0, the associated algorithm is made inactive.

Format

```
SP_STATUS SP_API RNBOSproDecrement (  
    RBP_SPRO_APIPACKET                thePacket,  
    RB_WORD                            writePassword,  
    RB_WORD                            address );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>writePassword</i>	IN	RB_WORD	The write password for the SuperPro key.
<i>address</i>	IN	RB_WORD	The address of the counter to decrement.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

An error will be returned if:

- You try to decrement a locked or hidden word.
- Word at the address is not a counter or AC = 0.
- The counter is already 0.
- The write password is incorrect.

RNBOsproQuery

This function is used to query an active algorithm at the specified address.

The query data pointer will point to the first byte of the data to be passed to the algorithm. The length of the query data (in bytes) is specified in the length variable. The minimum length is 4 bytes and the maximum length is 56 bytes.

On success, the query response will be placed in the buffer pointed to by the response pointer. It will have the same length as the query data. The last four bytes of the query response will also be placed in the response32 variable.

Note: It is the programmer's responsibility to allocate the memory for the buffers.

Format

```
SP_STATUS SP_API RNBOsproQuery (
    RBP_SPRO_APIPACKET          thePacket ,
    RB_WORD                     address ,
    RBP_VOID                    queryData ,
    RBP_VOID                    response ,
    RBP_DWORD                   response32 ,
    RB_WORD                     length );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>address</i>	IN	RB_WORD	The address of the word to query. It must point to the first word of an active algorithm.
<i>queryData</i>	IN	RBP_VOID	The pointer to the first byte of the query bytes.
<i>response</i>	OUT	RBP_VOID	The pointer to the first byte of the response bytes.
<i>response32</i>	OUT	RBP_DWORD	The pointer to the location that will contain a copy of the last four bytes of the query response.
<i>length</i>	IN	RB_WORD	This is the number of query bytes to send to the active algorithm and also the length of the response buffer.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section "API Error Codes" on page 37.

Comments

If the address is not the first word of an active algorithm, the return status will be successful and the response buffer data will be the same as the query buffer data.

RNBOsproActivate

This function activates an inactive algorithm at the specified cell address.

Format

```
SP_STATUS SP_API RNBOsproActivate (
    RBP_SPRO_APIPACKET    thePacket,
    RB_WORD                writePassword,
    RB_WORD                activatePassword1,
    RB_WORD                activatePassword2,
    RB_WORD                address );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>writePassword</i>	IN	RB_WORD	The write password for the key.
<i>activatePassword1</i>	IN	RB_WORD	The first word of the activation password.
<i>activatePassword2</i>	IN	RB_WORD	The second word of the activation password.
<i>address</i>	IN	RB_WORD	The address of the first word of an inactive algorithm.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- You can call this function anytime after obtaining a license.
- An error will be returned if:
 - The write password is invalid.
 - The activation password 1 or activation password 2 is invalid.
 - The cell address does not point to the word 1 of the algorithm.

RNBOsproGetSubLicense

This function obtains a sublicense from a locked data word (has an access code 1).

Format

```
SP_STATUS SP_API RNBOsproGetSubLicense (
    RBP_SPRO_APIPACKET          thePacket,
    RB_WORD                      address );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>address</i>	IN	RB_WORD	The address of a locked data word from which the sublicense will be obtained.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

You can call the RNBOsproGetSubLicense function only after calling the RNBOsproFindFirstUnit function.

Note: The key's hard limit is decremented first, then the sublicense limit is decremented for the requested application.

RNBOsproGetKeyType

This function returns the following information about the key attached to a system:

- **Key Family**
The key family parameter will return 0 or 1, where 0 denotes the SuperPro keys (the SSP keys) and 1 denotes the UltraPro keys (the SUP keys).
- **Form Factor**
The form factor parameter will return 0 or 1, where 0 denotes the parallel keys and 1 denotes the USB keys.
- **Memory Size**
The number of cells (inclusive of the reserved cells).

Format

```
SP_STATUS SP_API RNBOsproGetKeyType (
    RBP_SPRO_APIPACKET thePacket,
    RBP_WORD            KeyFamily,
    RBP_WORD            KeyFormFactor,
    RBP_WORD            KeyMemorySize );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>KeyFamily</i>	OUT	RBP_WORD	A pointer to an integer value that represents the key's family.
<i>KeyFormFactor</i>	OUT	RBP_WORD	A pointer to an integer value that represents the key's form factor.
<i>KeyMemorySize</i>	OUT	RBP_WORD	A pointer to an integer value that represents the number of cells in the key.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

You can call this function anytime after obtaining a license.

RNBOsproGetHardLimit

This function retrieves the hard limit of the key from which the license was obtained.

Format

```
SP_STATUS SP_API RNBOsproGetHardLimit (
    RBP_SPRO_APIPACKET    thePacket,
    RBP_WORD               hardLimit );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>hardLimit</i>	OUT	RBP_WORD	A pointer to the location that holds the hard limit of the key. It defines the maximum number of licenses that can be issued by this key.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

You can call this function anytime after obtaining a license.

RNBOsproGetVersion

This function returns the Sentinel driver's version and type.

Format

```
SP_STATUS SP_API RNBOsproGetVersion (
    RBP_SPRO_APIPACKET thePacket,
    RBP_BYTE majVer,
    RBP_BYTE minVer,
    RBP_BYTE rev,
    RBP_BYTE osDrvrType );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>majVer</i>	OUT	RBP_BYTE	A pointer to the location for the major version number returned.
<i>minVer</i>	OUT	RBP_BYTE	A pointer to the location for the minor version number returned.
<i>rev</i>	OUT	RBP_BYTE	A pointer to the location for the revision number returned.
<i>osDrvrType</i>	OUT	RBP_BYTE	<p>A pointer to the location where the operating system driver type information is stored. Currently defined types are:</p> <ul style="list-style-type: none"> <input type="checkbox"/> DOS local driver <input type="checkbox"/> Windows 3.x local driver <input type="checkbox"/> Windows Win32s local driver <input type="checkbox"/> Windows 3.x system driver <input type="checkbox"/> Windows NT system driver <input type="checkbox"/> OS/2 system driver <input type="checkbox"/> Windows 95 system driver <input type="checkbox"/> NetWare local driver <input type="checkbox"/> QNX local driver <input type="checkbox"/> UNIX local driver <input type="checkbox"/> SOLARIS local driver <input type="checkbox"/> Linux system driver <input type="checkbox"/> Linux local driver <input type="checkbox"/> AIX system driver <input type="checkbox"/> UNIX system driver

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section "API Error Codes" on page 37.

Comments

You can call this function anytime after obtaining a license.

RNBosproGetFullStatus

This function obtains the return code of the last-called API function.

Format

```
SP_STATUS SP_API RNBosproGetFullStatus (  
    RBP_SPRO_APIPACKET thePacket );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.

Return Code

If successful, the function returns the status of the last-called API function. If an error occurs, the function returns one of the error codes listed in the section “API Error Codes” on page 37.

Comments

- This function is provided for support purposes only. It returns an RB_WORD value that can be interpreted by the Technical Support.
- You can call this function anytime after obtaining a license.

RNBOsproGetKeyInfo

This function retrieves the following information about the key attached on a stand-alone system or a network computer (where the Sentinel Protection Server is running):

- Developer ID
- Hard limit
- Licenses in-use
- Licenses timed-out
- Highest number of licenses used

Format

```
SP_STATUS SP_API RNBOsproGetKeyInfo (
    RBP_SPRO_APIPACKET          thePacket,
    RB_WORD                     devId,
    RB_WORD                     keyIndex,
    NSPRO_MONITOR_INFO          *nsproMonitorInfo );
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>devId</i>	IN	RBP_WORD	If 0xFFFF is specified, the function will return the developer ID of the key along with other information.
<i>keyIndex</i>	IN	RBP_WORD	The index of the key whose information is sought. <ul style="list-style-type: none">❑ For cascaded parallel port keys: The sequential position of the key in the queue.❑ For multiple USB Keys: The order in which the key is plugged into the USB port/hub.
<i>*nsproMonitorInfo</i>	OUT	RBP_WORD	A pointer to the nsproMonitorInfo structure. This structure has various fields that contain information about the key. Refer to <i>spromeps.h</i> for details.

Return Code

If an error occurs, the function returns one of the error codes listed in the section “API Error Codes” on page 37.

Comments

You can call this function anytime after obtaining a license.

RNBOsproEnumServer

This function enumerates the number of Sentinel Protection Servers running in the subnet for the developer ID specified.

Format

```
SP_STATUS SP_API RNBOsproEnumServer (
    ENUM_SERVER_FLAG          enumFlag,
    RB_WORD                   developerId,
    NSPRO_SERVER_INFO         *serverInfo,
    RBP_WORD                   numServerInfo );
```

Parameters

Name	Direction	Parameter Type	Description
<i>enumFlag</i>	IN	ENUM_SERVER_FLAG	The flag used for contacting any of the following: <ul style="list-style-type: none"> ❑ NSPRO_RET_ON_FIRST_AVAILABLE (first-found Sentinel Protection Server that has a license to offer). ❑ NSPRO_RET_ON_FIRST (first-found Sentinel Protection Server that may have a license). ❑ NSPRO_GET_ALL_SERVERS (all the Sentinel Protection Servers in the subnet).
<i>developerId</i>	IN	RB_WORD	The developer ID of the SuperPro key to find. The Sentinel Protection Servers running on the system having a key of matching developer ID ONLY will respond. If developer ID is specified as 0xFFFF, then all the Sentinel Protection Servers (for a specified protocol) will respond.
<i>*serverInfo</i>	OUT	NSPRO_SERVER_INFO	A pointer to a buffer that will contain the Sentinel Protection Server information, such as the system address and the number of licenses available. A developer needs to allocate memory for the buffer.
<i>numServerInfo</i>	IN/OUT	RBP_WORD	A pointer to a variable that contains the desired number of the Sentinel Protection Servers. When the function returns, this variable contains the actual number of Sentinel Protection Servers found running on the network.

Return Code

If an error occurs, the function returns one of the error codes listed in the section “API Error Codes” on page 37.

Comments

You can call this function anytime after obtaining a license.

RNBOsproReleaseLicense

This function can be used to either release a license or sublicense(s).

Format

```
SP_STATUS SP_API RNBOsproReleaseLicense(
    RBP_SPRO_APIPACKET thePacket,
    RB_WORD address,
    RBP_WORD numSubLic);
```

Parameters

Name	Direction	Parameter Type	Description
<i>thePacket</i>	IN	RBP_SPRO_APIPACKET	A pointer to the API packet defined on page 4.
<i>address</i>	IN	RB_WORD	Specify zero to release the main license. Else, specify the cell address to release the sublicense from a particular cell.
<i>numSubLic</i>	IN/OUT	RBP_WORD	The pointer to the variable containing the number of sublicenses to be released. If the main license is to be released, this can be specified as null.

Return Code

If successful, the function returns SP_SUCCESS. If an error occurs, the function returns one of the codes listed in the section “API Error Codes” on page 37.

Comments

- You can call this function anytime after obtaining a license; followed by RNBOsproCleanUp.
- You can call this function before your application terminates. For example, the handler of the exit command button in your user interface can make use of this function.

We recommend you to use this function in order to release the idle licenses for other clients in queue. This function is especially useful in cases where you have set the heartbeat interval as infinite. The Sentinel Protection Server will not release the license unless you call this function.

RNBOSproCleanUp

This function releases the memory resources acquired by the SuperPro client library.

Format

```
RB_VOID SP_API RNBOSproCleanUp ( );
```

Comments

You can call this function immediately after calling RNBOSproReleaseLicense.

Data Type, Constants and Structure Definitions

This section provides information about the data types, constants and structures used in this document.

Data Type Definitions

<code>typedef unsigned long int*</code>	<code>RBP_DWORD;</code>
<code>typedef unsigned long int</code>	<code>RB_DWORD;</code>
<code>typedef unsigned short int*</code>	<code>RBP_WORD;</code>
<code>typedef unsigned short int</code>	<code>RB_WORD;</code>
<code>typedef unsigned short int</code>	<code>SP_STATUS;</code>
<code>typedef unsigned short int</code>	<code>PROTOCOL_FLAG;</code>
<code>typedef long int</code>	<code>RB_LONG;</code>
<code>typedef unsigned char*</code>	<code>RBP_BYTE;</code>
<code>typedef unsigned char</code>	<code>RB_BYTE;</code>
<code>typedef unsigned char</code>	<code>RB_BOOLEAN;</code>
<code>typedef char</code>	<code>RB_CHAR;</code>
<code>typedef void</code>	<code>RB_VOID;</code>
<code>typedef RB_VOID SP_PTR</code>	<code>RBP_VOID;</code>
<code>typedef RB_BYTE SP_PTR</code>	<code>RBP_BYTE;</code>
<code>typedef RB_BOOLEAN SP_PTR</code>	<code>RBP_BOOLEAN;</code>
<code>typedef RB_CHAR SP_PTR</code>	<code>RBP_CHAR;</code>

Constants

<code>#define SP_SUCCESS</code>	<code>0</code>
<code>#define _RB_API</code>	<code>__stdcall</code>
<code>#define SP_API</code>	<code>_RB_API</code>
<code>#define _RB_PTR</code>	<code>*</code>
<code>#define SP_PTR</code>	<code>_RB_PTR</code>
<code>#define MAX_ADDR_LEN</code>	<code>32</code>
<code>#define MAX_NAME_LEN</code>	<code>64</code>

```
❑ #define KEY_FORM_FACTOR_PARALLEL 0
❑ #define KEY_FORM_FACTOR_USB      1
❑ #define SSP_FAMILY_KEY           0
❑ #define SSP_E_FAMILY_KEY         1
```

Enumeration Flag Definition

```
/* Flags to specify the way of enumerating the Sentinel Protection Servers */  
  
#define NSPRO_RET_ON_FIRST          1  
  
#define NSPRO_GET_ALL_SERVERS      2  
  
#define NSPRO_RET_ON_FIRST_AVAILABLE 4
```

Protocol Flag Definition

```
/*To set the communication protocol flags*/  
  
typedef RB_WORD PROTOCOL_FLAG;  
  
#define NSPRO_TCP_PROTOCOL          1  
  
#define NSPRO_IPX_PROTOCOL          2  
  
#define NSPRO_NETBEUI_PROTOCOL      4  
  
#define NSPRO_SAP_PROTOCOL          8
```

Access Modes Definition

```
/*To set an access modes for the protected application*/  
  
#define RNBO_STANDALONE              "RNBO_STANDALONE"  
  
#define RNBO_SPN_DRIVER              "RNBO_SPN_DRIVER"  
  
#define RNBO_SPN_LOCAL               "RNBO_SPN_LOCAL"  
  
#define RNBO_SPN_BROADCAST           "RNBO_SPN_BROADCAST"  
  
#define RNBO_SPN_ALL_MODES           "RNBO_SPN_ALL_MODES"  
  
#define RNBO_SPN_SERVER_MODES       "RNBO_SPN_SERVER_MODES"
```

Heartbeat Definition

```
/*To make the license update time programmable*/  
  
#define LIC_UPDATE_INT              120          /* Default heartbeat 2 min */  
  
#define MAX_HEARTBEAT               2592000     /* 30*24*60*60 seconds */  
  
#define MIN_HEARTBEAT               60          /* 60 seconds */  
  
#define INFINITE_HEARTBEAT          -1          /* For infinite heartbeat */
```

Monitoring Information Structure Definition

```
/*Information about the key, used as part of the tag_nsproMonitorInfo
structure*/
typedef struct tag_nsproKeyMonitorInfo {
    RB_WORD          devId;
    RB_WORD          hardLimit;
    RB_WORD          inUse;
    RB_WORD          numTimeOut;
    RB_WORD          highestUse;
} NSPRO_KEY_MONITOR_INFO;

/*Information of the Sentinel Protection Server with the key details*/
typedef struct tag_nsproMonitorInfo {
    char              serverName[MAX_NAME_LEN];
    char              serverIPAddress[MAX_ADDR_LEN];
    char              serverIPXAddress[MAX_ADDR_LEN];
    char              version[MAX_NAME_LEN];
    RB_WORD           protocol;
    NSPRO_KEY_MONITOR_INFO sproKeyMonitorInfo;
} NSPRO_MONITOR_INFO;

/*The Sentinel Protection Server information with the number of
licenses available*/
typedef struct {
    char              serverAddress[MAX_ADDR_LEN];
    RB_WORD           numLicAvail;
} NSPRO_SERVER_INFO;
```

API Error Codes

This section contains a list of all recoverable API error codes. The documented errors are all recoverable errors. If you receive any unknown error numbers, please report the error number (extended error number if possible) to the Technical Support using the information provided on page 40.

Error Code (Decimal)	Description
0	SP_SUCCESS The function completed successfully.
1	SP_INVALID_FUNCTION_CODE You specified an invalid function code. See the include file for your language/interface (for example, spomeps.h) for valid API function codes. Generally, this error should not occur if you are using an interface provided by us to communicate with the Sentinel Driver. However, it may occur when a stand-alone "only" function is used in a network situation.
2	SP_INVALID_PACKET A checksum error was detected in the command packet, indicating an internal inconsistency. The packet record has not been initialized, or may have been tampered with. Generally, this error should not occur if you are using an interface provided by us to communicate with the Sentinel driver.
3	SP_UNIT_NOT_FOUND Unable to find the desired hardware key. Please verify if the key has been attached properly. Make sure you are sending the correct parameters.
4	SP_ACCESS_DENIED You attempted to perform an illegal action on a cell. For example, you may have tried to read an algorithm word, write to a locked cell, or decrement a cell that is not a data or counter word.
5	SP_INVALID_MEMORY_ADDRESS You specified an invalid memory address. You cannot operate on the reserved cells.
6	SP_INVALID_ACCESS_CODE You specified an invalid access code. The access code must be 0 (read/write data), 1 (read-only data), 2 (counter), 3 (algorithm/hidden), or 7 (AES algorithm).
7	SP_PORT_IS_BUSY The requested operation could not be completed because the port is busy. This can occur if there is considerable printer activity, or if a unit on the port is performing a write operation and is blocking the port. Try the function again.
8	SP_WRITE_NOT_READY The write or decrement operation could not be performed due to lack of sufficient power. Try the function again.
9	SP_NO_PORT_FOUND No parallel ports could be found, or there was a problem with the protocol being used on the network.
10	SP_ALREADY_ZERO You tried to decrement a counter that contains the value zero.
12	SP_ERR_DRIVER_NOT_INSTALLED The Sentinel driver was not installed or detected. Communication to the hardware key was not possible. Verify the device driver is correctly installed.

Error Code (Decimal)	Description
13	SP_IO_COMMUNICATIONS_ERROR The system device driver is having problems communicating. Verify the device driver is correctly installed.
15	SP_PACKET_TOO_SMALL The memory allocated for the API packet is less than the required size.
16	SP_INVALID_PARAMETER Arguments and values passed to the API function are invalid.
18	SP_VERSION_NOT_SUPPORTED The current system device driver is outdated. Update the driver.
19	SP_OS_NOT_SUPPORTED The operating system or environment is not supported by the client library. Contact Technical Support for assistance.
20	SP_QUERY_TOO_LONG You sent a query string longer than 56 characters. Send a shorter string.
21	SP_INVALID_COMMAND An invalid command was specified in the API call.
30	SP_DRIVER_IS_BUSY The Sentinel driver is busy. Try the function again.
31	SP_PORT_ALLOCATION_FAILURE Failure to allocate a parallel port through the operating system's parallel port contention handler.
32	SP_PORT_RELEASE_FAILURE Failure to release a previously allocated parallel port through the operating system's parallel port contention handler.
39	SP_ACQUIRE_PORT_TIMEOUT Failure to access the parallel port within the defined time.
42	SP_SIGNAL_NOT_SUPPORTED The particular system does not support a signal line. For example, an attempt may have been made to use the ACK line on an NEC 9800 computer.
57	SP_INIT_NOT_CALLED The key is not initialized.
58	SP_DRIVER_TYPE_NOT_SUPPORTED The type of driver access, either direct I/O or system driver, is not supported for the defined operating system and client library.
59	SP_FAIL_ON_DRIVER_COMM The client library failed to communicate with the Sentinel driver.
60	SP_SERVER_PROBABLY_NOT_UP The Sentinel Protection Server is not responding or the client has timed-out.
61	SP_UNKNOWN_HOST The Sentinel Protection Server host is unknown or not on the network, or an invalid host name was specified.
62	SP_SENDTO_FAILED The client was unable to send a message to the Sentinel Protection Server.

Error Code (Decimal)	Description
63	SP_SOCKET_CREATION_FAILED Client was unable to open a network socket. Make sure the TCP/IP or IPX protocol stack is properly installed on the system.
64	SP_NORESOURCES Could not locate enough licensing requirements. Insufficient resources (such as memory) are available to complete the request.
65	SP_BROADCAST_NOT_SUPPORTED Broadcast is not supported by the network interface on the system.
66	SP_BAD_SERVER_MESSAGE Could not understand the message received from the Sentinel Protection Server. An error occurred in decrypting (or decoding) the message at the client-end.
67	SP_NO_SERVER_RUNNING Cannot communicate to the Sentinel Protection Server. It may not be available for processing the license request on the specified host. Verify if the Sentinel Protection Server is running on the system.
68	SP_NO_NETWORK Unable to talk to the specified host. Network communication problems encountered.
69	SP_NO_SERVER_RESPONSE There is no Sentinel Protection Server running in the subnet, or the desired key is not available.
70	SP_NO_LICENSE_AVAILABLE All licenses are currently in use. The key has no more licenses to issue.
71	SP_INVALID_LICENSE License is no longer valid. It probably expired due to time-out.
72	SP_INVALID_OPERATION Specified operation cannot be performed. Probably, you tried setting the Sentinel Protection Server after obtaining a license, or tried to call the RNBOsproFindFirstUnit function on a packet that already has a license.
73	SP_BUFFER_TOO_SMALL The size of the buffer is not sufficient to hold the expected data.
74	SP_INTERNAL_ERROR An internal error, such as failure to encrypt or decrypt a message being sent or received, has occurred.
75	SP_PACKET_ALREADY_INITIALIZED The API packet was already initialized.
76	SP_PROTOCOL_NOT_INSTALLED The specified protocol is not installed.
104	SP_NO_DIGITAL_SIGNATURE The Sentinel driver binary is not signed by a valid authority.
105	SP_SYS_FILE_CORRUPTED The digital certificate of the Sentinel driver is not valid.

Contacting Technical Support

We are committed to supporting Sentinel SuperPro. If you have questions, need additional assistance, or encounter a problem, please contact our Technical Support using one of the methods listed in the following table:

http://www.safenet-inc.com/support/index.asp	
Americas	
Internet	http://www.safenet-inc.com/support/index.asp
E-mail	techsupport@safenet-inc.com
United States	
Telephone	(800) 545-6608
Fax	(949) 450-7450
Europe	
E-mail	eutechsupport@safenet-inc.com
France	
Telephone	0825 341000
Fax	+44 (0) 1276 608080
Germany	
Telephone	01803 7246269
Fax	+44 (0) 1276 608080
United Kingdom	
Telephone	+44 (0) 1276 608000
Fax	+44 (0) 1276 608080

Pacific Rim	
E-mail	techsupportpacrim@safenet-inc.com
Australia and New Zealand	
Telephone	(61) 3 9882 8322
Fax	(61) 3 9882 0588
China	
Telephone	(86) 10 8851 9191
Fax	(86) 10 6872 7342
India	
Telephone	(91) 11 2691 7538
Fax	(91) 11 2633 1555
Taiwan and Southeast Asia	
Telephone	(886) 2 27353736
Fax	(886) 2 27352383

Copyright © 2005, SafeNet, Inc. (Baltimore)

All rights reserved.

<http://www.safenet-inc.com>

SafeNet, Sentinel, SuperPro, and UltraPro are either registered trademarks or trademarks of SafeNet, Inc. Microsoft, Windows, Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP, Windows Server 2003 and Internet Explorer are either trademarks or registered trademarks of Microsoft Corporation in the United States and other countries. All other trademarks are the property of their respective owners.

March 2005, Revision A

