

Sentinel SuperPro ***Microsoft Visual Basic Interface***



© Copyright 2002, Rainbow Technologies, Inc.
All rights reserved.
<http://www.rainbow.com>

All attempts have been made to make the information in this document complete and accurate. Rainbow Technologies, Inc. is not responsible for any direct or indirect damages or loss of business resulting from inaccuracies or omissions. The specifications contained in this document are subject to change without notice.

Sentinel SuperPro is a trademark of Rainbow Technologies, Inc. All other product names referenced herein are trademarks or registered trademarks of their respective manufacturers.

October, 2002

RAINBOW TECHNOLOGIES, INC.

50 Technology Drive, Irvine, CA 92618

Telephone: (949) 450-7300, (800) 852-8569 Fax: (949) 450-7450

RAINBOW TECHNOLOGIES LTD.

4 The Forum, Hanworth Lane, Chertsey, Surrey KT16 9JX, United Kingdom

Telephone: (44) 1932 579200 Fax: (44) 1932 570743

RAINBOW TECHNOLOGIES

122, Avenue Charles de Gaulle, 92522 Neuilly-sur-Seine Cedex, France

Telephone: (33) 1 41 43 29 02 Fax: (33) 1 46 24 76 91

RAINBOW TECHNOLOGIES GMBH

Streiflacher Str. 7, Germering, D 82110, Germany

Telephone: (49) 89 32 17 98 15 Fax: (49) 89 32 17 98 50

Additional offices and distributors are located worldwide.

International Quality Standard Certification

Rainbow Technologies, Inc. Irvine, CA facility has been issued the ISO 9001 certification, the globally recognized standard for quality, by Det Norske Veritas as of March 2002.
Certificate Number CERT-02982-2000-AQ-HOU-RABR2

Table of Contents

About This Document.....	vi
Conventions Used in This Document	vi
Suggested References	vi
Getting Help	vii
Interface Requirements	1
Compiler Compatibility	1
Specific Requirements	1
Testing	1
Build Example Information	1
Evaluation Program Files	1
Build Example Instructions	2
Sentinel SuperPro Interface APIs	2
The API Packet	2
RNBOsproFormatPacket.....	3
RNBOsproInitialize.....	3
RNBOsproSetProtocol	4
RNBOsproSetContactServer.....	4
RNBOsproFindFirstUnit.....	5
RNBOsproGetContactServer	6
RNBOsproSetHeartBeat	6
RNBOsproFindNextUnit	7
RNBOsproRead	7
RNBOsproExtendedRead	8
RNBOsproWrite	9
RNBOsproOverwrite.....	10
RNBOsproDecrement	11
RNBOsproActivate	12
RNBOsproQuery	13
RNBOsproGetVersion	14
RNBOsproGetHardLimit.....	15
RNBOsproGetKeyInfo	15
RNBOsproGetFullStatus	16
RNBOsproGetSubLicense	16
RNBOsproReleaseLicense	17
RNBOsproEnumServer	17
Data Types, Constants and Structure Definitions	18
Data Types Used in the Interface	18
Constants	19
Query Definition	20
Monitoring Information Structure Definition	20
Server Information Structure Definition	20
API Status Codes	21

About This Document

This document contains information on using the Sentinel SuperPro Microsoft Visual BASIC interface. It describes the interface requirements, build examples, SuperPro client library APIs and their status codes.

Conventions Used in This Document

Please note the following conventions regarding text this document:

Convention	Meaning
COURIER	Denotes syntax, prompts and code examples. If bold, denotes text you type.
<i>Italics</i>	Text in italics denotes the parameter names, file names and directories, or for emphasis in notes and tips.
Bold Lettering	In procedures, words in boldface type represent keystrokes, menu items, window names or mouse commands.

Suggested References

Refer to the following documentation for more detailed information about Sentinel SuperPro.

Document	What's in it ?
<i>Sentinel SuperPro 6.1 Developer's Guide.</i>	The detailed information about the product features and APIs.
<i>Sentinel SuperPro 6.3 Documentation Addendum</i>	Contains information about the product changes done since the 6.2 release.

Getting Help

If you have questions, need additional assistance, or encounter a problem, please contact Rainbow Technologies Technical Support using one of the methods listed in the following table:

Rainbow Technologies Technical Support Contact Information

Corporate Headquarters North America and South America	
	Rainbow Technologies North America
Internet	http://www.rainbow.com/support.html
E-mail	techsupport@irvine.rainbow.com
Telephone	(800) 959-9954 (Monday – Friday, 6:00 a.m. – 6:00 p.m. PST)
Fax	(949) 450-7450
Australia and New Zealand	
E-mail	techsupport@au.rainbow.com
Telephone	(61) 3 9820 8900
Fax	(61) 3 9820 8711
China	
E-mail	sentinel@isecurity.com.cn
Telephone	(86) 10 8266 3936
Fax	(86) 10 8266 3948
France	
E-mail	EuTechSupport@rainbow.com
Telephone	(33) 1 41.43.29.00
Fax	+44 (0) 1932 570743
Germany	
E-mail	EuTechSupport@rainbow.com
Telephone	0183 RAINBOW (7246269)
Fax	+44 (0) 1932 570743
Taiwan and Southeast Asia	
E-mail	techsupport@tw.rainbow.com
Telephone	(886) 2 2570-5522
Fax	(886) 2 2570-1988
United Kingdom	
E-mail	EuTechSupport@rainbow.com
Telephone	0870 7529200
Fax	+44 (0) 1932 570743
Countries not listed above	
Please contact your local distributor for assistance.	

Interface Requirements

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

Compiler Compatibility

This interface is compatible with the following compilers:

- Microsoft Visual Basic 5.0
- Microsoft Visual Basic 6.0

Specific Requirements

The interface also requires the following:

- Sentinel System Driver 5.41 or higher
- Sentinel SuperPro Server version 6.3 or higher (for network operations only; not required for cases where *direct-to-driver* communication takes place. Also note that in case of the *direct-to-driver* communication, the network APIs—RNBOsproGetSubLicense, RNBOsproSetProtocol and RNBOsproSetHeartBeat—will return an error whenever called.)

Testing

This interface has been tested on the following platforms:

- Windows 98
- Windows NT
- Windows 2000
- Windows ME
- Windows XP (32-bit)

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>FrmEnSvr.frm</i>	A Visual Basic form for EnumServer.
<i>FrmEnSvr.frx</i>	A binary file for <i>frmEnSvr.frm</i> .
<i>FrmGetKeyInfo.frm</i>	A Visual Basic form for GetKeyInfo.
<i>FrmGetKeyInfo.frx</i>	A binary file for <i>frmGetKeyInfo.frm</i> .

File Name	Description
<i>frmSetconserver.frm</i>	A Visual Basic form for SetContactServer.
<i>frmSetconserver.frx</i>	A binary file for the <i>frmSetconserver.frm</i> form.
<i>FrmSetProt.frm</i>	A Visual Basic form for the SetProtocol.
<i>FrmSetProt.frx</i>	A binary file for <i>frmSetProt.frm</i> .
<i>Rnbo.ico</i>	An icon file for <i>sproeval.exe</i> .
<i>SPROEVAL.frm</i>	The main form of the evaluation code.
<i>SPROEVAL.frx</i>	A binary file for <i>Sproeval.frm</i> .
<i>SPROEVAL.mak</i>	The make file used for building <i>sproeval.exe</i> .
<i>SPROEVAL.vbp</i>	A Visual Basic project file for the example program.
<i>SPROEVAL.vbw</i>	A Visual Basic workspace file for the example program.
<i>Spromeps.bas</i>	The declare statements file required by Visual Basic to interface with the DLL subroutines.
<i>SuperPro Microsoft VB Interface.pdf</i>	(this document).
<i>Sx32w.dll</i>	The Sentinel SuperPro dynamic link library .

Build Example Instructions

1. Open the *sproeval.vbp* project file in the Visual Basic 5.0/6.0 IDE.
2. Click the **Make sproeval.exe** option from the **File** menu to build the executable.

Tip! In your application you need to use the *spromeps.bas* and *sx32w.dll* files to call the *SuperPro APIs*.

Sentinel SuperPro Interface APIs

The API Packet

All APIs require an APIPACKET structure. It is an 4112-byte array whose internal structure is known only by the SuperPro client library. The SuperPro driver uses the data in the APIPACKET to communicate with the SuperPro key. A programmer should never modify the data in the APIPACKET.

Packet Definition

```
Type APIPACKET
    Data(API_PACKET_SZ - 1) As Byte
End Type
```

The following APIs are supported by this interface.

Note: The equivalent C interface APIs are also included to provide information about the parameters passed. For information on the structures, constants and data types used, refer to the section on “Data Type, Constant and Structure Definitions.”

RNBOSproFormatPacket

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

Note: This API must be called before any other RNBOSpro API.

Format

```
Function RNBOSproFormatPacket% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET, _  
     ByVal ApiPackSize As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
ApiPack	IN	APIPACKET	The API packet defined earlier.
ApiPackSize	IN	Integer	The size of the APIPACKET.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOSproInitialize

This API initializes the packet and sets the contact server, if anything is set in the NSP_HOST environment variable.

Format

```
Function RNBOSproInitialize% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET)
```

Parameters

Name	Direction	Parameter Type	Description
ApiPack	IN	APIPACKET	The API packet defined earlier.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproInitialize(RBP_SPRO_APIPACKET packet);
```

RNBOsproSetProtocol

This API registers the communication protocol of a client to the server. It is called after initializing the packet and before RNBOsproFindFirst API is called. If this API is not used, the default protocol setting remains TCP/IP.

This API will not work if the API packet already has a license; in that case it will return an SP_INVALID_OPERATION error code.

Format

```
Function RNBOsproSetProtocol% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET, _  
     ByVal ProtocolFlag As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>ProtocolFlag</i>	IN	Integer	The protocol chosen by the client for communication with the server. The valid values are: NSPRO_TCP_PROTOCOL = 1 NSPRO_IPX_PROTOCOL = 2 NSPRO_NETBEUI_PROTOCOL = 3 NSPRO_SAP_PROTOCOL = 8 [†]

[†] Service Advertising Protocol (SAP) is used for finding the key plugged in the Novell server through broadcast only.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproSetContactServer

This API is used to set the contact server for a particular API packet. The contact server can be set as RNBO_STANDALONE, RNBO_SPN_DRIVER, RNBO_SPN_LOCAL, RNBO_SPN_BROADCAST, RNBO_SPN_ALL_MODES, RNBO_SPN_SERVER_MODES or as an IP address, IPX address, NetBEUI name or the name of the machine.

This API will not work if the API packet already has a license; in that case it will return an SP_INVALID_OPERATION error code.

Format

```
Function RNBOsproSetContactServer% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET, _  
     ByVal srvr As String)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>srvr</i>	IN	String	Any of the following reserved strings: <ul style="list-style-type: none">▪ RNBO_STANDALONE▪ RNBO_SPN_DRIVER▪ RNBO_SPN_LOCAL▪ RNBO_SPN_BROADCAST▪ RNBO_SPN_ALL_MODES▪ RNBO_SPN_SERVER_MODES▪ no-net[†] Or, name of the contact server (Servername/IP address/IPX address ^{††})

[†] The no-net mode is deprecated. See the Sentinel SuperPro 6.3 Documentation Addendum for details.

^{††} The IPX address should be represented in the “xx-xx-xx-xx,xx-xx-xx-xx-xx-xx” format, for example 12-34-56-78,9A-BC-DE - F0-12-34.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproSetContactServer(RBP_SPRO_APIPACKET packet,  
                                           char *serverName );
```

RNBOsproFindFirstUnit

This API finds the first SuperPro key on the server with the specified developer ID and gets a license from the key. If RNBOsproFindFirstUnit is called with an API packet, which already has a license, then a SP_INVALID_OPERATION error code is returned.

Format

```
Function RNBOsproFindFirstUnit% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET, _  
     ByVal DEVELOPERID As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>DEVELOPERID</i>	IN	Integer	The developer ID of the SuperPro key.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproFindFirstUnit(RBP_SPRO_APIPACKET packet,
                                         RB_WORD developerID );
```

RNBOsproGetContactServer

This API is used to return the contact server set for a particular API packet.

Format

```
Function RNBOsproGetContactServer% Lib "Sx32w.dll"
(ApiPack As APIPACKET, _
ByVal srvr As String, _
ByVal strlen As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>srvr</i>	OUT	String	A buffer in which the server name is copied. A developer needs to ensure that the buffers have sufficient memory.
<i>strlen</i>	IN	Integer	The length of the buffer. The maximum length recommended is up to 64 bytes.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproGetContactServer(RBP_SPRO_APIPACKET packet,
                                         char *serverNameBuf,
                                         RB_WORD serverNameBufSz);
```

RNBOsproSetHeartBeat

This API customizes the heartbeat of a client. It has to be called only after RNBOsproFindFirst is called. It can be used in following ways:

1. To set an infinite heartbeat for a client by setting the time to INFINITE_HEARTBEAT. In this case, the server will not release the license acquired by a client until RNBOsproReleaseLicense is received by the server for this client.
2. To set the heartbeat to any value between MIN_HEARTBEAT to MAX_HEARTBEAT in multiples of 1 second.

If the API is not used, the default heartbeat setting is 120 seconds.

Format

```
Function RNBOsproSetHeartBeat% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET, _  
    ByVal heartbeat As Long)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>heartbeat</i>	IN	Long	A value that represents time in seconds.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproSetHeartBeat(RBP_SPRO_APIPACKET packet,  
                                        RB_DWORD heartbeatValue);
```

RNBOsproFindNextUnit

This API finds the next SuperPro key based on the developer ID maintained in the APIPACKET. This API should not be called, unless RNBOsproFindFirstUnit has returned a successful value or, if the licenses available with the contacted server are exhausted (SP_NO_LICENSE_AVAILABLE).

If this API returns success, the system will release the license obtained by RNBOsproFindFirstUnit API call and will contain the data for the next SuperPro key. If this API returns an error value, the APIPACKET will be marked invalid.

To re-initialize the APIPACKET, use RNBOsproFindFirstUnit and optionally, RNBOsproFindNextUnit depending upon the number of SuperPro keys found and the one your program accesses.

Format

```
Function RNBOsproFindNextUnit% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproFindNextUnit(RBP_SPRO_APIPACKET packet);
```

RNBOsproRead

This API reads a word at the specified address of the SuperPro key identified by the APIPACKET.

On success, the data variable will contain information from the SuperPro key. If the error code returned is SP_ACCESS_DENIED, an attempt was made to read a non-readable (algorithm) word. For security reasons, algorithm words cannot be read.

Format

```
Function RNBOsproRead% Lib "Sx32w.dll" _
    (ApiPack As APIPACKET, _
    ByVal address As Integer _
    Datum As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>address</i>	IN	Integer	The cell address to be read.
<i>Datum</i>	OUT	Integer	Contains the data read from the key.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproRead(RBP_SPRO_APIPACKET packet,
                               RB_WORD address,
                               RBP_WORD data );
```

RNBOsproExtendedRead

This API reads the word and access code at the specified address of the SuperPro key identified by the APIPACKET.

On success, the data variable will contain the information from the SuperPro key and the access code variable will contain the access code. If the error code returned is SP_ACCESS_DENIED, an attempt was made to read a non-readable (algorithm) word. For security reasons, algorithm words cannot be read.

Format

```
Function RNBOsproExtendedRead% Lib "Sx32w.dll" _
    (ApiPack As APIPACKET, _
    ByVal address As Integer, _
    Datum As Integer, _
    accessCode As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>address</i>	IN	Integer	The cell address to be read.
<i>Datum</i>	OUT	Integer	Contains the data read from the key.
<i>accessCode</i>	OUT	Integer	The associated access code returned.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproWrite

This API is used to write a word and its associated access code to the SuperPro key identified by the APIPACKET.

Writing to the SuperPro key requires a write password. The word data is placed in the data variable and its associated access code is placed in the access code variable.

On success, the data and its associated access code are written to the specified word on the SuperPro key. If error code is SP_ACCESS_DENIED, either the write password was incorrect or an attempt was made to write/overwrite a locked cell.

The write API can be used only to write/overwrite words with an access code of 0. To overwrite words with other access codes, use the RNBOsproOverwrite API.

Format

```
Function RNBOsproWrite% Lib "Sx32w.dll" _
    (ApiPack As APIPACKET, _
    ByVal wPass As Integer, _
    ByVal address As Integer, _
    ByVal Datum As Integer, _
    ByVal accessCode As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>wPass</i>	IN	Integer	The write password of the key.
<i>address</i>	IN	Integer	The cell address to be written.
<i>Datum</i>	OUT	Integer	Contains the word to write in the key.
<i>accessCode</i>	OUT	Integer	Contains the access code associated with the word to write.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproWrite(RBP_SPRO_APIPACKET packet,
                                RB_WORD writePassword,
                                RB_WORD address,
                                RB_WORD data,
                                RB_BYTE accessCode);
```

RNBOsproOverwrite

This API writes a word and its associated access code to the SuperPro key identified by the APIPACKET.

Overwriting to the SuperPro key requires the write and overwrite passwords. The word data is placed in the data variable and its associated access code is placed in the access code variable. On success, the data and its associated access code are written to the specified word on the SuperPro key. If the error code returned is SP_ACCESS_DENIED, the write password and/or the overwrite passwords were incorrect.

This API can be used to overwrite any word on the SuperPro key with an exception of the words at addresses 0-7.

Format

```
Function RNBOsproOverwrite% Lib "Sx32w.dll" _
    (ApiPack As APIPACKET, _
    ByVal wPass As Integer, _
    ByVal oPass1 As Integer, _
    ByVal oPass2 As Integer, _
    ByVal address As Integer, _
    ByVal Datum As Integer, _
    ByVal accessCode As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>wPass</i>	IN	Integer	The write password for the key.
<i>oPass1</i>	IN	Integer	The word 1 of the overwrite password.
<i>oPass2</i>	IN	Integer	The word 2 of the overwrite password.
<i>address</i>	IN	Integer	The cell address to be written.
<i>Datum</i>	IN	Integer	Contains the word to write in the key.
<i>accessCode</i>	IN	Integer	Contains the access code associated with the word to write.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproDecrement

This API decrements the counter at the specified address of the SuperPro key identified by the APIPACKET. If the API is successful, the counter is decremented by 1. Errors are returned if you try to decrement a locked or hidden word, the counter is already 0, the word at the address is not a counter or, the write password is incorrect.

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

The counter and associated algorithm can appear in the SuperPro as:

Address	Data
N - 2	Counter
N - 1	Counter
N	Algorithm Word 1
N + 1	Algorithm Word 2

If either or both counters exist, the counter is associated with the algorithm. This association will exist only for N = 0C, 14, 1C, 24, 2C, 34, 3C Hex.

An algorithm can have both an associated password and associated counters. The counters can be used to make the algorithm inactive and the password can be used to make the algorithm active. See RNBOsproActivate.

Format

```
Function RNBOsproDecrement% Lib "Sx32w.dll"
    (ApiPack As APIPACKET, _
     ByVal wPass As Integer, _
     ByVal address As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>wPass</i>	IN	Integer	The write password for the key.
<i>address</i>	IN	Integer	The cell address of the counter to decrement.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproDecrement (RBP_SPRO_APIPACKET packet,
                                     RB_WORD               writePassword,
                                     RB_WORD               address );
```

RNBOsproActivate

This API is used to activate an inactive algorithm at the specified address of the SuperPro key identified by the APIPACKET.

If the API is successful, the algorithm is made active. Errors are returned if the write password is invalid, the activate password is invalid, or the address is not word 1 of an algorithm having an activation password.

The algorithm and associated password will appear in the SuperPro as:

Address	Data
N	Algorithm Word 1
N + 1	Algorithm Word 2
N + 2	Activate Password 1
N + 3	Activate Password 1

The association will only exist for N = 08, 0C, 10, 14, 18, 1C, 20, 24, 28, 2C, 30, 34, 38, 3C Hex. An algorithm can have both an associated password and associated counters. The counters can be used to make an algorithm inactive and the password can be used to make an algorithm active. See RNBOsproDecrement.

Format

```
Function RNBOsproActivate% Lib "Sx32w.dll"
    (ApiPack      As APIPACKET, _
     ByVal wPass  As Integer, _
     ByVal aPass1 As Integer, _
     ByVal aPass2 As Integer, _
     ByVal address As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The APIPACKET defined earlier.
<i>wPass</i>	IN	Integer	The write password for the SuperPro key.
<i>aPass1</i>	IN	Integer	The first word of the activate password.
<i>aPass2</i>	IN	Integer	The second word of the activate password.
<i>address</i>	IN	Integer	The cell address of the first word of an algorithm to activate.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproQuery

This API is used to query an active algorithm at the specified address of the SuperPro key identified by the APIPACKET .

The address should be the first word of an active algorithm. The query variable will point to the first byte of the data to be passed to an active algorithm. The length of the query data is specified in the length variable. On success, the query response of the same length is placed in the buffer pointed by the response variable. The last 4 bytes of the response will also be placed in the unused variable.

Each query byte may contain any value varying from 0 to 255. Each response byte may also contain any value between 0-255. The length of the response will always be the same as the length of the query bytes. It is the programmer's responsibility to have buffers with sufficient memory.

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

Format

```
Function RNBOsproQuery% Lib "Sx32w.dll"
(ApiPack As APIPACKET, _
ByVal address As Integer, _
query As DATAQUERY, _
response As DATAQUERY, _
unused As Long, _
ByVal length As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The APIPACKET defined earlier.
<i>address</i>	IN	Integer	The cell address of the word to query.
<i>query</i>	IN	DATAQUERY	The query data sent to the key.
<i>response</i>	OUT	DATAQUERY	The response information received from the key.
<i>unused</i>	OUT	Long	The last 4 bytes of the query response.
<i>length</i>	IN	Integer	The number of query bytes to be sent to an active algorithm and also the length of the response buffer.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproGetVersion

This API returns the driver's version number and type.

Format

```
Function RNBOsproGetVersion% Lib "Sx32w.dll" _
    (ApiPack As APIPACKET, _
    majv As Integer, _
    minv As Integer, _
    rev As Integer, _
    ostype As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>majv</i>	OUT	Integer	The major version number returned.
<i>minv</i>	OUT	Integer	The minor version number returned.
<i>rev</i>	OUT	Integer	The revision level returned.
<i>ostype</i>	OUT	Integer	The operating system driver type. Currently defined types are: <ol style="list-style-type: none">1. DOS local driver2. Windows 3.x local driver3. Windows Win32s local driver4. Windows 3.x system driver5. Windows NT system driver6. OS/2 system driver7. Windows 95 system driver8. NetWare local driver9. QNX local driver

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproGetHardLimit

This API is used to retrieve the maximum number of licenses supported by a key (the hard limit).

Format

```
Function RNBOsproGetHardLimit% Lib "Sx32w.dll" _  
    (ApiPack As APIPACKET, _  
     hrdLmt As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>hrdLmt</i>	OUT	Integer	A buffer that will hold the hard limit.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

```
SP_STATUS SP_API RNBOsproGetHardLimit(RBP_SPRO_APIPACKET packet,  
                                       RBP_WORD HardLimit );
```

RNBOsproGetKeyInfo

This API is used to get information about a key from a particular SuperPro server.

Format

```
Function RNBOsproGetKeyInfo% Lib "Sx32w.dll" _  
    (ApiPack As APIPACKET, _  
     ByVal devID As Long, _  
     ByVal keyIndex As Integer, _  
     monitorInfo As NSPRO_MONITOR_INFO)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>devID</i>	IN	Long	The developer ID of the key at the position specified by the keyIndex parameter.
<i>keyIndex</i>	IN	Integer	The index of the key whose information is being sought.
<i>monitorInfo</i>	OUT	NSPRO_MONITOR_INFO	A reference to the NSPRO_MONITOR_INFO structure. This structure has various fields that provide information about the key.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproGetFullStatus

This function is used for obtaining the return code of the last called API. It is provided for support purposes only.

Format

```
Function RNBOsproGetFullStatus% Lib "Sx32w.dll" _
                                     (ApiPack As APIPACKET)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.

Return Code

Returns an integer value that can be interpreted by Rainbow's technical support department.

Equivalent C Interface API

```
RB_WORD SP_API RNBOsproGetFullStatus(RBP_SPRO_APIPACKET packet );
```

RNBOsproGetSubLicense

This API is used to get a sublicense from the read-only data cell.

Format

```
Function RNBOsproGetSubLicense% Lib "Sx32w.dll" _
                                     (ApiPack      As APIPACKET, _
                                     ByVal cellAdd As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>cellAdd</i>	IN	Integer	The address of the cell to get the sublicense from.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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


RNBOsproReleaseLicense

This API can be used in two ways:

1. To release the main license by specifying the cell address as zero.
2. To release the sublicense from a particular cell by specifying the cell address of the sublicensing cell as well as the number of sublicenses to be released.

Format

```
Function RNBOsproReleaseLicense% Lib "Sx32w.dll"  
    (ApiPack As APIPACKET, _  
     ByVal cellAdd As Integer, _  
     numSubLic As Long)
```

Parameters

Name	Direction	Parameter Type	Description
<i>ApiPack</i>	IN	APIPACKET	The API packet defined earlier.
<i>cellAdd</i>	IN	Integer	The cell address of the sublicense. If a sublicense is to be released, specify the sublicense cell number, otherwise specify zero.
<i>numSubLic</i>	IN/OUT	Long	A variable containing the number of sublicenses to be released. If the main license is to be released, this can be specified as zero.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

RNBOsproEnumServer

This API is used to enumerate the number of SuperPro servers running in a subnet for the developer ID specified.

Format

```
Function RNBOsproEnumServer% Lib "Sx32w.dll"  
    (ByVal enumFlag As Integer, _  
     ByVal devID As Long, _  
     serverInfo As SrvrInfoArr, _  
     numServers As Integer)
```

Parameters

Name	Direction	Parameter Type	Description
<i>enumFlag</i>	IN	Integer	The flag used for contacting either the; - first-found SuperPro server that has licenses to offer (NSPRO_RET_ON_FIRST_AVAILABLE), - or the first-found SuperPro server that may have licenses (NSPRO_RET_ON_FIRST), - or all the SuperPro servers in the network (NSPRO_GET_ALL_SERVERS).
<i>devID</i>	IN	Long	The developer ID for the SuperPro device to find. Only the servers that have a key matching the developer ID will respond. If developer ID is specified as &HFFFF, then all servers (for a specified protocol) in the subnet will respond.
<i>serverInfo</i>	OUT	SrvrInfoArr	A buffer that contains the SuperPro server information, such as the server address and the number of licenses available. A developer needs to ensure that the buffer has sufficient memory.
<i>numServers</i>	IN/OUT	Integer	An in-out parameter that contains the desired number of SuperPro servers to be enumerated. When the function returns, it contains the number of servers that were actually enumerated.

Return Code

On success, returns SP_SUCCESS. Else, returns an error code as defined in the “API Status Codes” section at the end of this document.

Equivalent C Interface API

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

Data Types, Constants and Structure Definitions

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

Data Types Used in the Interface

Definition	Data Type in C	Data Type in Visual Basic
RB_DWORD	unsigned long int	Long
ENUM_SERVER_FLAG	int	Integer
SP_STATUS	unsigned short int	--
PROTOCOL_FLAG	unsigned short int	--
RB_BYTE	unsigned char	byte
RBP_DWORD	unsigned long int*	Long

Definition	Data Type in C	Data Type in Visual Basic
RBP_WORD	unsigned short int*	--
RBP_BYTE	unsigned char*	--
RBP_VOID	void*	--

-- Not Applicable

Constants

- Global Const MAX_NAME_LEN = 64
- Global Const MAX_ADDR_LEN = 32
- Global Const SPRO_MAX_QUERY_SIZE = 56
- Global Const API_PACKET_SZ = 4112
- Global Const MAX_NUM_SERVERS = 10

Protocol Constants

` Set the protocol flags

```
Global Const NSPRO_TCP_PROTOCOL = 1
Global Const NSPRO_IPX_PROTOCOL = 2
Global Const NSPRO_NETBEUI_PROTOCOL = 4
Global Const NSPRO_SAP_PROTOCOL = 8
```

Heartbeat Constants

`Making the license update time programmable

```
Global Const MAX_HEARTBEAT = 2592000 '30*24*60*60 seconds
Global Const MIN_HEARTBEAT = 60 '60 seconds
Global Const INFINITE_HEARTBEAT = &HFFFFFFFF 'For infinite heartbeat
```

Enumeration Flag Constants

`The enum server flags

```
Global Const NSPRO_RET_ON_FIRST = 1
Global Const NSPRO_GET_ALL_SERVERS = 2
Global Const NSPRO_RET_ON_FIRST_AVAILABLE = 4
```

Access Modes Definition

`To set an access modes for the protected application

```
Global Const RNBO_STANDALONE = "RNBO_STANDALONE"
Global Const RNBO_SPN_DRIVER = "RNBO_SPN_DRIVER"
Global Const RNBO_SPN_LOCAL = "RNBO_SPN_LOCAL"
Global Const RNBO_SPN_BROADCAST = "RNBO_SPN_BROADCAST"
Global Const RNBO_SPN_ALL_MODES = "RNBO_SPN_ALL_MODES"
Global Const RNBO_SPN_SERVER_MODES = "RNBO_SPN_SERVER_MODES"
```

Query Definition

```
Type DATAQUERY
  Data (SPRO_MAX_QUERY_SIZE - 1) As Byte
End Type
```

Monitoring Information Structure Definition

'Key-related information used in the tag_nsproMonitorInfo structure

```
Type NSPRO_KEY_MONITOR_INFO
  devID      As Integer
  hrdLmt     As Integer
  LicInUse   As Integer
  NumTimedOut As Integer
  highestUse As Integer
End Type
```

'The SuperPro server information with key details

```
Type NSPRO_MONITOR_INFO
  SrvrName(MAX_NAME_LEN - 1) As Byte
  srvrIPAdd(MAX_ADDR_LEN - 1) As Byte
  srvrIPXAdd(MAX_ADDR_LEN - 1) As Byte
  version(MAX_NAME_LEN - 1) As Byte
  protocol As Integer
  keyInfo As NSPRO_KEY_MONITOR_INFO
End Type
```

Server Information Structure Definition

'The SuperPro server information with the number of licenses available

```
Type NSPRO_SERVER_INFO
  srvrAdd(MAX_ADDR_LEN - 1) As Byte
  numLicAvail As Integer
End Type

Type SrvrInfoArr
  srvrInfo(MAX_NUM_SERVERS - 1) As NSPRO_SERVER_INFO
End Type
```

API Status Codes

On success, all APIs discussed earlier return SP_SUCCESS. Else, they return an error code defined below. This section enumerates all the recoverable API status codes with their description. However, if you receive any unknown error codes, please report it to Rainbow Technologies Technical Support.

Note: API Status Codes not listed below are obsolete even though they appear in sprome.ps.bas.

Status Code (Decimal)	Description
0	SP_SUCCESS The function completed successfully.
1	SP_INVALID_FUNCTION_CODE An invalid function code was specified. See your language's include file for valid API function codes. Generally, this error should not occur if you are using a Rainbow-provided interface to communicate with the driver.
2	SP_INVALID_PACKET A checksum error was detected in the command packet, indicating an internal inconsistency. The packet structure may have been tampered with. Generally, this error should not occur if you are using a Rainbow-provided interface to communicate the driver.
3	SP_UNIT_NOT_FOUND The specific unit could not be found. Make sure you are sending the correct information to find the unit. This error is returned by other functions if the unit has disappeared or unplugged.
4	SP_ACCESS_DENIED You attempted to perform an illegal action on a word. For example, you may have tried to read an algorithm/hidden word, write to a locked word, or decrement a word that is not a data nor a counter word.
5	SP_INVALID_MEMORY_ADDRESS You specified an invalid Sentinel SuperPro memory address. Valid addresses are 0-63 decimal(0-3F hex). Cells 0-7 are invalid for many operations. Algorithm descriptors must be referenced by the first (even) address.
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), or 3 (algorithm/hidden).
7	SP_PORT_IS_BUSY The port is busy in some other operation.
8	SP_WRITE_NOT_READY The write or decrement action could not be performed due to a momentary lack of sufficient power. Attempt the operation again.
9	SP_NO_PORT_FOUND No ports could be found on the workstation.
10	SP_ALREADY_ZERO You tried to decrement a counter or data word that already contains the value 0. If you are using the counter to control demo program executions, this condition may occur after corresponding algorithm descriptor has been reactivated with its activation password.
12	SP_DRIVER_NOT_INSTALLED The system device driver was not installed or detected. Communication with the unit was not possible. Please verify that the device driver is properly loaded.

Status Code (Decimal)	Description
13	SP_IO_COMMUNICATIONS_ERROR The system device driver is having problems communicating with the unit. Please verify that the device driver is properly installed.
15	SP_PACKET_TOO_SMALL The API packet is too small.
16	SP_INVALID_PARAMETER The API packet contained an invalid parameter.
18	SP_VERSION_NOT_SUPPORTED The current system device driver is outdated. Please update the system device driver.
19	SP_OS_NOT_SUPPORTED The Operating System or environment is currently not supported by the client library. Please contact Rainbow Technical Support.
20	SP_QUERY_TOO_LONG The maximum length of a query string supported is 56 characters. Retry with a shorter string.
21	SP_INVALID_COMMAND An invalid SuperPro command was specified in the API call.
30	SP_DRIVER_IS_BUSY The system device driver is busy. Try the operation again.
31	SP_PORT_ALLOCATION_FAILURE Failed to allocate a parallel port through the Operating System's parallel port contention handler.
32	SP_PORT_RELEASE_FAILURE Failed to release a previously allocated parallel port through the Operating System's parallel port contention handler.
39	SP_ACQUIRE_PORT_TIMEOUT Failed to acquire access to a parallel port within the defined time-out.
42	SP_SIGNAL_NOT_SUPPORTED The particular machine does not support a signal line. For example, an attempt was made to use the ACK line on a NEC 9800 computer. The NEC 9800 does not have an ACK line. Therefore, this error is reported.
57	SP_INIT_NOT_CALLED Failed to call the client library's initialize API. This API must be called prior to the API that generated this error.
58	SP_DRV_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 on communicating with a Rainbow system driver.
60	SP_SERVER_PROBABLY_NOT_UP Server is not responding and the client has been timed out.
61	SP_UNKNOWN_HOST Unknown server host. Server host does not seem to be on the network. Invalid hostname.
62	SP_SENDTO_FAILED Client was unable to send message to the server.
63	SP_SOCKET_CREATION_FAILED Client was unable to open network socket. Make sure the TCP/IP or IPX protocol stack is properly installed on the machine.

Status Code (Decimal)	Description
64	SP_NORESOURCES Could not locate enough licensing resources. Insufficient resources (such as memory) are available to complete the request. An error occurred in attempting to allocate memory needed by function.
65	SP_BROADCAST_NOT_SUPPORTED Broadcast is not supported by the network interface on the machine.
66	SP_BAD_SERVER_MESSAGE Could not understand message received from the server. An error occurred in decrypting(or decoding) a server message at the client end.
67	SP_NO_SERVER_RUNNING Cannot talk to the server. Verify server is running. No server seems to be running. Server on specified host is not available for processing the client request.
68	SP_NO_NETWORK Unable to talk to the specified host. Network communication problems encountered.
69	SP_NO_SERVER_RESPONSE No server responded to client broadcast. Either there is no server running in the subnet or no server in the subnet has a desired key attached. Also can be the case, when a particular server (the contact server for that client) is not responding back.
70	SP_NO_LICENSE_AVAILABLE All licenses are currently in use. Server has no more licenses available for this request.
71	SP_INVALID_LICENSE The license is no longer valid. License expired due to timeout.
72	SP_INVALID_OPERATION The specified operation cannot be performed. A license has already been issued for the given APIPACKET. Trying to set contact server after obtaining a license for the given APIPACKET, or trying to make another findfirst call.
73	SP_BUFFER_TOO_SMALL The size of the buffer is not sufficient to hold the expected data.
74	SP_INTERNAL_ERROR Internal error faced in licensing.
75	SP_PACKET_ALREADY_INITIALIZED The given APIPACKET has already been initialized.
76	SP_PROTOCOL_NOT_INSTALLED The protocol specified is not installed.