Sentinel SuperPro
Borland Delphi Interface

RAINBOW
TECHNOLOGIES

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

This document contains information on using the Sentinel SuperPro Delphi interface. It describes the interface requirements, build example, Sentinel SuperPro library APIs and API 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.
Italics	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 on Sentinel SuperPro.

Document	What's in it ?
Sentinel SuperPro 6.1 Developer's Guide.	For detailed information about the product features and APIs.
Sentinel SuperPro 6.3 Documentation Addendum	Contains information about the product changes since 6.1.1 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:

- Borland Delphi 5.0
- Borland Delphi 6.0

Specific Requirements

The interface requires the following:

- Sentinel System Driver 5.41 or higher
- Sentinel SuperPro Server version 6.3 or higher (for network operations only; not required when *direct-to-driver* communication takes place. Further, 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
Activate.dfm	The binary file for the activate form.
Activate.pas	The source code for the Activate form.
EnumServer.dfm	The binary file for the EnumServer form.

File Name	Description
EnumServer.pas	The source code for the EnumServer form.
Eval.dfm	The binary file for the main form.
Eval.pas	The source code of the main form.
Key_info.dfm	The binary file for the GetKeyInfo form.
Key_info.pas	The source code for the GetKeyInfo form.
Mkeval.bat	A batch file used for building *sproeval.exe*.
Password.dfm	The binary file for the password form.
Password.pas	The source code for the example program.
Rnbo.ico	The icon file used in the example.
SPROEVAL.dpr	The Delphi project file for the example program.
SPROEVAL.res	The Delphi compiled resource file for the example program.
Spromeps.lib	The Sentinel SuperPro static link library.
Spromeps.pas	The source code that implements the calls to the SuperPro APIs.
Sx32w.dll	The Sentinel SuperPro dynamic link library.
Sx32w.lib	The Sentinel SuperPro import library for *sx32w.dll.*
SuperPro Borland Delphi Interface.pdf	(this document)

***Note**: Borland Kylix interface is also included in this release of Sentinel SuperPro.*

Build Example Instructions

To build the *sproeval.exe* program, follow the steps given below:

1. Open the *sproeval.dpr* project file in the Borland Delphi 5.0 or 6.0 IDE.
2. Click **Build All Projects** from the **Project** menu to build the executable.
3. Now, run *sproeval.exe* from its destination.

***Tip!** In your application you need to use the following files to call the SuperPro APIs:*
- spromeps.pas
- spromeps.lib *or* sx32w.dll *(and* sx32w.lib*)*

Sentinel SuperPro Interface APIs

The RB_SPRO_APIPACKET Record

All functions require an APIPACKET record. It is an 1028-char array whose internal structure is defined in the SuperPro client library. The library uses the data in the APIPACKET to communicate with the SuperPro key. A programmer should never modify the data in an APIPACKET.

Packet Definition

```
RB_SPRO_APIPACKET     = ARRAY [0..RB_SPRO_APIPACKET_SIZE] OF CHAR;
RB_SPRO_APIPACKET_PTR = ^RB_SPRO_APIPACKET;
```

***Note**: The equivalent C interface APIs are also included to provide information about the parameters passed.*

SproFormatPacket

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

***Note**: This API must be called before any other spro function.*

Format

```
FUNCTION SproFormatPacket(ApiPacket:  RB_SPRO_APIPACKET_PTR;
                          PacketSize: WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
PacketSize	IN	WORD	The size of the RB_SPRO_APIPACKET record.

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 );
```

SproInitialize

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

Format

```
FUNCTION SproInitialize(ApiPacket: RB_SPRO_APIPACKET_PTR): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET 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);
```

SproSetProtocol

This function registers the communication protocol of a client with the SuperPro server. This function is called after initializing the packet and before sproFindFirst API is called. If this function 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 SproSetProtocol(ApiPacket:    RB_SPRO_APIPACKET_PTR;
                         ProtocolFlag: WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	Pointer to the RB_SPRO_APIPACKET defined earlier.
ProtocolFlag	IN	WORD	The protocol chosen by a 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);
```

SproSetContactServer

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 SproSetContactServer(ApiPacket:  RB_SPRO_APIPACKET_PTR;
                              ServerName: PChar): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
ServerName	IN	PChar	Any of the following reserved strings: ▪ 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.*

† † 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);
```

SproFindFirstUnit

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

Format

```
FUNCTION SproFindFirstUnit(ApiPacket:   RB_SPRO_APIPACKET_PTR;
                           DeveloperID: WORD ): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
developerID	IN	WORD	The developer ID of the SuperPro device to find.

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 );
```

SproGetContactServer

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

Format

```
FUNCTION SproGetContactServer(ApiPacket:     RB_SPRO_APIPACKET_PTR;
                              ServerName:    PChar;
                              ServerNameLen: WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
ServerName	OUT	PChar	A buffer in which the server name is copied.
ServerNameLen	IN	WORD	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);
```

SproSetHeartBeat

This function customizes the heartbeat of a client. It has to be called only after sproFindFirst is called. It can be used to:

1. 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 sproReleaseLicense 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 function is not used, the default heartbeat setting remains is seconds.

Format

```
FUNCTION SproSetHeartBeat(ApiPacket: RB_SPRO_APIPACKET_PTR;
                          Heartbeat: DWORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	Pointer to the RB_SPRO_APIPACKET defined earlier.
Heartbeat	IN	DWORD	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);
```

SproFindNextUnit

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

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

To re-initialize the RB_SPRO_APIPACKET record, use sproFindFirstUnit and optionally, sproFindNextUnit depending upon the number of SuperPro keys found and the one your program accesses.

Format

```
FUNCTION SproFindNextUnit(ApiPacket:RB_SPRO_APIPACKET_PTR): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET 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);
```

SproRead

This function reads a word at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET record. On success, the data variable will contain information from the SuperPro key. If SP_ACCESS_DENIED error code is returned, an attempt was made to read a non-readable (algorithm) word. For security reasons, algorithm words cannot be read.

Format

```
FUNCTION SproRead(ApiPacket:     RB_SPRO_APIPACKET_PTR;
                  Address:       WORD;
                  data:POINTER): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
address	IN	WORD	The cell address to be read.
data	OUT	POINTER	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 );
```

SproExtendedRead

This function reads the word and access code at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET record. On success, the data variable will contain the information from the SuperPro key and the access code variable will contain the access code. If SP_ACCESS_DENIED error

code is returned, an attempt was made to read a non-readable (algorithm) word. For security reasons, algorithm words cannot be read.

Format

```
FUNCTION SproExtendedRead(ApiPacket:  RB_SPRO_APIPACKET_PTR;
                          Address:    WORD;
                          Data:       POINTER;
                          AccessCode: POINTER): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
address	IN	WORD	The cell address to be read.
data	OUT	POINTER	Contains the data read from the key.
accessCode	OUT	POINTER	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 );
```

SproWrite

This function is used to write a word and its associated access code to the SuperPro key identified by the RB_SPRO_APIPACKET record.

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 SP_ACCESS_DENIED error code is returned, either the write password was incorrect or an attempt was made to write/overwrite a locked cell.

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

Format

```
FUNCTION SproWrite(ApiPacket:     RB_SPRO_APIPACKET_PTR;
                   WritePassword: WORD;
                   address:       WORD;
                   data:          WORD;
                   accessCode:    WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
writePassword	IN	WORD	The write password of the key.
address	IN	WORD	The cell address to be written.
data	IN	WORD	Contains the word to write in the key.
accessCode	IN	WORD	Contains an 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 );
```

SproOverwrite

This function writes a word and its associated access code to the SuperPro key identified by the RB_SPRO_APIPACKET record.

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 *accessCode* variable. On success, the data and its associated access code are written to the specified word on the SuperPro key. If the error code is SP_ACCESS_DENIED, the write password and/or the overwrite passwords were incorrect. This function can be used to overwrite any word on the SuperPro key with an exception of the words at addresses 0-7.

Format

```
FUNCTION SproOverwrite(ApiPacket:          RB_SPRO_APIPACKET_PTR;
                       writePassword:      WORD;
                       overwritePassword1: WORD;
                       overwritePassword2: WORD;
                       address:            WORD;
                       data:               WORD;
                       accessCode: WORD):  WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
writePassword	IN	WORD	The write password of the key.
overwritePassword1	IN	WORD	The word 1 of the overwrite password.
overwritePassword2	IN	WORD	The word 2 of the overwrite password.

Name	Direction	Parameter Type	Description
address	IN	WORD	The cell address to be written.
data	IN	WORD	Contains the word to write in the key.
accessCode	IN	WORD	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 );
```

SproDecrement

This function decrements the counter at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET record. If the function 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 sproActivate.

Format

```
FUNCTION SproDecrement(ApiPacket:      RB_SPRO_APIPACKET_PTR;
                       writePassword:  WORD;
                       address: WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.

Name	Direction	Parameter Type	Description
writePassword	IN	WORD	The write password of the key.
address	IN	WORD	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);
```

SproActivate

This function is used to activate an inactive algorithm at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET record. If the function 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 the address of word 1 of an algorithm having an activate 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 sproDecrement.

Format

```
FUNCTION SproActivate(ApiPacket:          RB_SPRO_APIPACKET_PTR;
                      writePassword:      WORD;
                      activatePassword1: WORD;
                      activatePassword2: WORD;
                      address: WORD):     WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier
writePassword	IN	WORD	The write password of the key.
activatePassword1	IN	WORD	The first word of the activate password.
activatePassword2	IN	WORD	The second word of the activate password.

Name	Direction	Parameter Type	Description
address	IN	WORD	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);
```

SproQuery

This function is used to query an active algorithm at the specified address of the SuperPro key identified by the RBP_SPRO_APIPACKET record.

The address should be the first word of an active algorithm. The query data 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 response32 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 allocate memory for the buffers.

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 SproQuery(ApiPacket:     RB_SPRO_APIPACKET_PTR;
                   Address:       WORD;
                   QueryData:     POINTER;
                   Response:      POINTER;
                   response32:    LONGINT;
                   length: WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
Address	IN	WORD	The cell address of the word to query.
QueryData	IN	POINTER	A pointer to the first byte of the query bytes.
Response	OUT	POINTER	A pointer to the first byte of the response bytes.
response32	OUT	LONG	A variable that contains a copy of the last 4 bytes of the query response.
length	IN	WORD	The number of bytes in query data.

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);
```

SproGetVersion

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

Format

```
FUNCTION SproGetVersion(ApiPacket:      RB_SPRO_APIPACKET_PTR;
                        VAR majVer:     BYTE;
                        VAR minVer:     BYTE;
                        VAR rev:        BYTE;
                        VAR osDrvrType: BYTE): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.
majVer	OUT	BYTE	The major version number returned.
minVer	OUT	BYTE	The minor version number returned.
rev	OUT	BYTE	The revision level returned.
osDrvrType	OUT	BYTE	The operating system driver type. Currently defined types are: 1. DOS local driver 2. Windows 3.x local driver 3. Windows Win32s local driver 4. Windows 3.x system driver 5. Windows NT system driver 6. OS/2 system driver 7. Windows 95 system driver 8. NetWare local driver 9. 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);
```

SproGetHardLimit

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

Format

```
FUNCTION SproGetHardLimit(ApiPacket: RB_SPRO_APIPACKET_PTR;
                          HardLmt:   POINTER): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	Pointer to the RB_SPRO_APIPACKET defined earlier.
HardLmt	OUT	Pointer	Buffer of length 1 to hold the hard limit. Memory should be allocated by a developer.

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);
```

SproGetKeyInfo

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

Format

```
FUNCTION SproGetKeyInfo( ApiPacket:   RB_SPRO_APIPACKET_PTR ;
                         devID:       WORD ;
                         KeyIndex:    WORD;
                         MonitorInfo: NSPRO_MONITOR_INFO_PTR ) : WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	Pointer to the RB_SPRO_APIPACKET record.
devId	IN	WORD	Developer ID of the key at the position specified by the *KeyIndex* parameter.
KeyIndex	IN	WORD	Index of the key with developer ID has to be provided.

Name	Direction	Parameter Type	Description
monoInfo	OUT	NSPRO_MONITOR_INFO_PTR	Pointer to the NSPRO_MONITOR_INFO structure. Memory needs to be allocated by a developer for this structure. This structure has various fields which 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);
```

SproGetFullStatus

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

Format

```
FUNCTION SproGetFullStatus( ApiPacket: RB_SPRO_APIPACKET_PTR ): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	A pointer to the RB_SPRO_APIPACKET defined earlier.

Return Code

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

Equivalent C Interface API

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

SproGetSubLicense

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

Format

```
FUNCTION SproGetSubLicense(ApiPacket: RB_SPRO_APIPACKET_PTR;
                           Address:   WORD): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	Pointer to the RB_SPRO_APIPACKET defined earlier.

Name	Direction	Parameter Type	Description
Address	IN	WORD	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);
```

SproReleaseLicense

This function 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 SproReleaseLicense(ApiPacket: RB_SPRO_APIPACKET_PTR;
                            Address  : WORD;
                            NumSubLic: POINTER): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
ApiPacket	IN	RB_SPRO_APIPACKET_PTR	The RB_SPRO_APIPACKET defined earlier.
Address	IN	Word	Cell address of the sublicense. If a sublicense is to be released, specify the sublicense cell number, otherwise specify 0.
NumSubLic	IN/OUT	Pointer	Pointer to a 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

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);
```

SproEnumServer

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

Format

```
FUNCTION SproEnumServer(enumFlag:   Integer;
                        devID:      WORD ;
                        serverInfo: NSPRO_SERVER_INFO_PTR;
                        numServer:  POINTER): WORD;
```

Parameters

Name	Direction	Parameter Type	Description
enumFlag	IN	Integer	The flag used for contacting the, - first found server that has licenses to offer (NSPRO_RET_ON_FIRST_AVAILABLE), - or first found server that may have licenses or not (NSPRO_RET_ON_FIRST), - or all the servers in the network (NSPRO_GET_ALL_SERVERS).
devID	IN	WORD	The developer ID for the SuperPro device to find. Only the servers that have a key matching the developer ID will respond. If the developer ID is specified as $FFFF then all the servers (for a specified protocol) in the subnet would respond.
serverInfo	OUT	NSPRO_SERVER_INFO_PTR	Pointer to the NSPRO_SERVER_INFO structure. Buffer that will contain the server information, such as the server address and the number of licenses available with the server for each server that responded. User needs to allocate memory for the buffer.
numServer	IN/OUT	Pointer	Pointer to a variable that contains the desired number of servers. When function returns, this variable contains the actual number of servers found on the network.

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 Type, Constant and Structure Definitions

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

Data Types Defined in C Interface

Definition	Data Type in C	Data Type in Delphi (Pascal)
RB_DWORD	unsigned long int	LongInt

Definition	Data Type in C	Data Type in Delphi (Pascal)
ENUM_SERVER_FLAG	int	Integer
SP_STATUS	unsigned short int	WORD
PROTOCOL_FLAG	unsigned short int	Integer
RB_BYTE	unsigned char	BYTE
RBP_DWORD	unsigned long int*	^LongInt
RBP_WORD	unsigned short int*	^WORD
RBP_BYTE	unsigned char*	PChar
RBP_VOID	void*	POINTER

-- Not Applicable

Constants

- MAX_NAME_LEN = 64;
- MAX_ADDR_LEN = 32;

Monitoring Information Structure Definition

```
{Structure for KeyInfo API}
  type
     NSPRO_KEY_MONITOR_INFO = record
       devId      :WORD;
       hardLimit  :WORD;
       inUse      :WORD;
       numTimeOut :WORD;
       highestUse :WORD;
     end;

{Structure for Monitoring info}
  type
     NSPRO_MONITOR_INFO = record
       serverName       : ARRAY [0..MAX_NAME_LEN - 1] OF CHAR;
       serverIPAddress  : ARRAY [0..MAX_ADDR_LEN - 1] OF CHAR;
       serverIPXAddress : ARRAY [0..MAX_ADDR_LEN - 1] OF CHAR;
       version          : ARRAY [0..MAX_NAME_LEN - 1] OF CHAR;
       protocol         : WORD;
     end;
     NSPRO_MONITOR_INFO_PTR = ^NSPRO_MONITOR_INFO;
```

Server Enumeration Information

Server Structure Definition

```
{Structure for EnumSerevr}
  type
        NSPRO_SERVER_INFO = record
           serverAddress  :  ARRAY [0..MAX_ADDR_LEN - 1] OF CHAR;
           numLicAvail    :  WORD;
        end;
        NSPRO_SERVER_INFO_PTR = ^NSPRO_SERVER_INFO;
```

Enumeration Flag Definition

```
{Enum server flags}
NSPRO_RET_ON_FIRST           = 1;
NSPRO_GET_ALL_SERVERS        = 2;
NSPRO_RET_ON_FIRST_AVAILABLE = 4;
```

Protocol Flag Definition

```
{To set the communication protocol flags}
NSPRO_TCP_PROTOCOL           = 1;
NSPRO_IPX_PROTOCOL           = 2;
NSPRO_NETBEUI_PROTOCOL       = 4;
NSPRO_SAP_PROTOCOL           = 8;
```

Heartbeat Definition

```
{To make the license update time programmable}
MAX_HEARTBEAT                = 2592000;
MIN_HEARTBEAT                = 60;
INFINITE_HEARTBEAT           = $FFFFFFFF;
```

Access Modes Definition

```
(To set the access modes for the protected application)

RNBO_STANDALONE       := 'RNBO_STANDALONE';
RNBO_SPN_LOCAL        := 'RNBO_SPN_LOCAL';
RNBO_SPN_DRIVER       := 'RNBO_SPN_DRIVER';
RNBO_SPN_BROADCAST    := 'RNBO_SPN_BROADCAST';
RNBO_SPN_ALL_MODES    := 'RNBO_SPN_ALL_MODES';
RNBO_SPN_SERVER_MODES := 'RNBO_SPN_SERVER_MODES';
```

API Status Codes

On success, all functions 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 numbers, please report the error number (extended error number if possible) to Rainbow Technologies Technical Support.

***Note**: The API Status Codes not listed below are obsolete even though they appear in* spromeps.pas.

Status Code (Decimal)	Description
0	**SP_SUCCESS** The fuction 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.

Status Code (Decimal)	Description
2	**SP_INVALID_PACKET** A checksum error was detected in the command packet, indicating an internal inconsistency. The packet structure 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 date), 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 prgram 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.
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.

Status Code (Decimal)	Description
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_DRVR_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.
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 communcation 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.

Status Code (Decimal)	Description
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.