Sentinel SuperPro AutoCAD 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 AutoCAD interface. It describes the interface requirements, build examples, 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 the 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:

- AutoDesk AutoCAD 2000
- AutoDesk AutoCAD 2002

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 for cases where direct-to-driver communication takes place. Also note that in case of 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)

***Note**: AutoCAD 2000 is not supported on the Windows XP (32-bit) platform.*

Build Example Information

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

Evaluation Program Files

File Name	Description
Resource.h	An include file used by *sproeval.rc.*
.\ARX16\SPROARX.arx	An ObjectARX file used as glue layer (built with ARX16)

File Name	Description
.\ARX2002\SPROARX.arx	An ObjectARX file used as glue layer (built with ARX2002)
Build.bat	A batch file used for *sproARX.arx*.
SproARX.cpp	The source code for *sproARX.arx*.
sproARX.DEF	The definition file for *sproARX.arx*.
sproARX.dep	The dependency file for *sproARX.arx*.
sproARX.dsp	The Microsoft Visual C++ 6.0 project file for *sproARX.arx*.
sproARX.dsw	The Microsoft Visual C++ 6.0 project workspace for *sproARX.arx*.
Sproeval.lsp	The LISP file for the AutoCAD example program.
sproARX.mak	The make File used for building the glue layer (*sproARX.arx*).
Sprometps.h	An include file for the Sentinel SuperPro APIs.
Sprometps.lib	The Sentinel SuperPro static link library built with multi-threaded (MT) flag.
VERSION.rc	A resource file that contains the version information.
SuperPro AutoCAD Interface.pdf	(this document)

Build Example Instructions

You can compile and run the example program using the recommended compiler. Follow the instructions given below:

1. Open the AutoCAD 2000/2002 development environment.
2. Click **Load Application** under the **Tools** menu option, then load the *sproARX.arx* available under *.\ARX16* or.*\ARX2002* respectively and *sproeval.lsp* files.
3. At the AutoCAD command prompt, type **`(sproeval)`** and press the **Enter** key to build and run the example program.

Tip! *In your application you need to use the appropriate* SPROARX.arx *file (built with ARX16 or ARX2002) to call the SuperPro APIs.*

Sentinel SuperPro Interface APIs

The SuperPro APIs are implemented in C language. The AutoCAD compiler does not support direct calls to C/C++ functions. Hence, a glue layer is built using ObjectARX of AutoCAD to provide an interface between the LISP code and the Sentinel SuperPro library.

In the glue layer wrappers are defined for each SuperPro API. These wrappers need to be registered by the same name or different name so that a LISP application (AutoCAD interface) can call them.

All the glue layer wrappers return output as a linked list of type *resbuf* struct, as defined in ObjectARX. *Resbuf* is a general-purpose result buffer structure that handles all the AutoCAD data types.

The RB_SPRO_APIPACKET Structure

Most of the SuperPro APIs require a pointer to the RB_SPRO_APIPACKET structure. The glue layer takes care of this structure internally and does not require a prgrammer to pass it externally from the LISP application (AutoCAD interface)†. The SuperPro driver uses the data in the RB_SPRO_APIPACKET structure to communicate with the SuperPro key.

Packet Definition in C

```
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 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 RB_SPRO_APIPACKET (in the glue layer itself) based on its size. It must be called before any other function.

Registered Name

```
sproFormatPacket
```

Glue Layer Wrapper

```
ADS_FormatPacket
```

Prototype

```
RB_WORD ADS_FormatPacket (void);
```

Inputs

None

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

† The distict nature of LISP implementation prohibits from adding support for vectors, hashes or any other convenient data structures such as structs or objects, destructive modifications on lists, other language improvements such as macros.

RNBOsproInitialize

This function initializes the packet.

Registered Name

```
sproInitialize
```

Glue Layer Wrapper

```
ADS_Initialize
```

Prototype

```
RB_WORD ADS_Initialize (void);
```

Inputs

None

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproSetProtocol

This function registers the communication protocol of a client with the SuperPro server. This function is called after initializing the packet and before RNBOsproFindFirst 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.

Registered Name

```
sproSetProtocol
```

Glue Layer Wrapper

```
ADS_SetProtocol
```

Prototype

```
RB_WORD ADS_SetProtocol (char *protocolFlag);
```

Inputs

Name	Direction	Parameter Type	Description
protocolFlag	IN	char*	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.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproSetContactServer

This API is used to set the SuperPro server to be contacted 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.

Registered Name

```
sproSetContactServer
```

Glue Layer Wrapper

```
ADS_SetContactServer
```

Prototype

```
RB_WORD ADS_SetContactServer (char *serverName);
```

Inputs

Name	Direction	Parameter Type	Description
serverName	IN	char*	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, the 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.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproFindFirstUnit

This function finds the first key on the SuperPro 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 is returned.

Registered Name

```
sproFindFirstUnit
```

Glue Layer Wrapper

```
ADS_FindFirstUnit
```

Prototype

```
RB_WORD ADS_FindFirstUnit (char *Dev_id);
```

Inputs

Name	Direction	Parameter Type	Description
Dev_id	IN	char*	The developer ID of the SuperPro key to find.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproGetContactServer

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

Registered Name

```
sproGetContactServer
```

Glue Layer Wrapper

```
ADS_GetContactServer
```

Prototype

```
RB_WORD ADS_GetContactServer (void);
```

Inputs

None

Output

Gives a list of type *resbuf* containing the return code of the API and the server name.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproSetHeartBeat

This function 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 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 function is not used, the default heartbeat setting is 120 seconds.

Registered Name

```
sproSetHeartBeat
```

Glue Layer Wrapper

```
ADS_SetHeartBeat
```

Prototype

```
RB_WORD ADS_SetHeartBeat (char *heartBeat);
```

Inputs

Name	Direction	Parameter Type	Description
heartBeat	IN	char*	A value that represents time in seconds.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproFindNextUnit

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

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

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

Registered Name

```
sproFindNextUnit
```

Glue Layer Wrapper

```
ADS_FindNextUnit
```

Prototype

```
RB_WORD ADS_FindNextUnit (void);
```

Inputs

None

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproRead

This function reads a word at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET structure. On success, the resbuf list 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.

Registered Name

```
sproRead
```

Glue Layer Wrapper

```
ADS_Read
```

Prototype

```
RB_WORD ADS_Read (char *address);
```

Inputs

Name	Direction	Parameter Type	Description
address	IN	char*	The cell address to be read.

Output

Gives a list of type *resbuf* containing the return code of the API and the data read.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproExtendedRead

This function reads the word and access code at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET structure. On success, the resbuf list will contain the information from the SuperPro key and 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.

Registered Names

```
sproExtendedRead
```

Glue Layer Wrapper

```
ADS_ExtendedRead
```

Prototype

```
RB_WORD ADS_ExtendedRead (char *address);
```

Inputs

Name	Direction	Parameter Type	Description
address	IN	char*	The cell address to be read.

Output

Gives a list of type *resbuf* containing the return code of the API, data read and access code of the word read.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 RB_SPRO_APIPACKET structure.

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 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.

Registered Name

```
sproWrite
```

Glue Layer Wrapper

```
ADS_Write
```

Prototype

```
RB_WORD ADS_Write (char          *writePassword,
                   char          *address,
                   char          *data,
                   unsigned char  accessCode);
```

Inputs

Name	Direction	Parameter Type	Description
writePassword	IN	char*	The write password of the key.
address	IN	char*	The cell address to be written.
data	IN	char*	Contains the word to write in the key.
accessCode	IN	unsigned char	Contains an access code associated with the word to write.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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

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 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.

Registered Name

```
sproOverwrite
```

Glue Layer Wrapper

```
ADS_Overwrite
```

Prototype

```
RB_WORD ADS_Overwrite (char          *writePassword,
                       char          *overwritePassword1,
                       char          *overwritePassword2,
                       char          *address,
                       char          *data,
                       unsigned char  accessCode);
```

Inputs

Name	Direction	Parameter Type	Description
writePassword	IN	char*	The write password of the key.
overwritePassword1	IN	char*	The word 1 of the overwrite password.
overwritePassword2	IN	char*	The word 2 of the overwrite password.
address	IN	char*	The cell address to be written..
data	IN	char*	Contains the word to write in the key.
accessCode	IN	unsigned char	Contains the access code associated with the word to write.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 function decrements the counter at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET structure. 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 RNBOsproActivate.

Registered Name

```
sproDecrement
```

Glue Layer Wrapper

```
ADS_Decrement
```

Prototype

```
RB_WORD ADS_Decrement (char   *writePassword ,
                      char   *address);
```

Inputs

Name	Direction	Parameter Type	Description
writePassword	IN	char*	The write password of the key.
address	IN	char*	The cell address of the counter to decrement.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproActivate

This function is used to activate an inactive algorithm at the specified address of the SuperPro key identified by the RB_SPRO_APIPACKET structure. 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 RNBOsproDecrement.

Registered Name

```
sproActivate
```

Glue Layer Wrapper

```
ADS_Activate
```

Prototype

```
RB_WORD ADS_Activate (char  *writePassword,
                      char  *activatePassword1,
                      char  *activatePassword2,
                      char  *address);
```

Inputs

Name	Direction	Parameter Type	Description
writePassword	IN	char*	The write password of the key.
activatePassword1	IN	char*	The first word of the activate password.
activatePassword2	IN	char*	The second word of the activate password.
address	IN	char*	The cell address of the first word of an algorithm to activate.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 function is used to query an active algorithm at the specified address of the SuperPro key identified by the RBP_SPRO_APIPACKET structure. 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. On success, the query response is returned.

Each query byte may contain any value varying from 0 to 255. Each response byte may also contain any value between 0-255.

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

Registered Name

```
sproQuery
```

Glue Layer Wrapper

```
ADS_Query
```

Prototype

```
RB_WORD ADS_Query (char   *address,
                   char   *queryData);
```

Inputs

Name	Direction	Parameter Type	Description
address	IN	char*	The cell address of the word to query.
queryData	IN	char*	The query data sent to the key.

Output

Gives a list of type *resbuf* containing the return code of the API and query response returned.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 function returns the driver's version number and type.

Registered Name

```
sproGetVersion
```

Glue Layer Wrapper

```
ADS_GetVersion
```

Prototype

```
RB_WORD ADS_GetVersion (void);
```

Inputs

None

Output

Returns a list of type *resbuf* containing the return code of the API and the major version, minor version, revision and operating system driver type.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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

Registered Name

```
sproGetHardLimit
```

Glue Layer Wrapper

```
ADS_GetHardLimit
```

Prototype

```
RB_WORD ADS_GetHardLimit (void);
```

Inputs

None

Output

Gives a list of type *resbuf* containing the return code and the hard limit of the key.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 attached on a stand-alone system or a network computer.

Registered Name

```
sproGetKeyInfo
```

Glue Layer Wrapper

```
ADS_GetKeyInfo
```

Prototype

```
RB_WORD ADS_GetKeyInfo (char *keyID,
                        char *keyIndex);
```

Inputs

Name	Direction	Parameter Type	Description
keyID	IN	char*	The developer ID of the key at the position specified by the *keyIndex* parameter.
keyIndex	IN	char*	The index of the key whose information is being sought.

Output

Gives a list of type *resbuf* containing the return code of the API, developer ID, hard limit, number of licenses in use, number of licenses timed out and maximum number of license issued against a key.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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.

Registered Name

```
sproGetFullStatus
```

Glue Layer Wrapper

```
ADS_GetFullStatus
```

Prototype

```
RB_WORD ADS_GetFullStatus (void);
```

Inputs

None

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

Equivalent C Interface API

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

RNBOsproGetSubLicense

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

Registered Name

```
sproGetSubLicense
```

Glue Layer Wrapper

```
ADS_GetSubLicense
```

Prototype

```
RB_WORD ADS_GetSubLicense (char *address);
```

Inputs

Name	Direction	Parameter Type	Description
address	IN	char*	The cell address to get the sublicense from.

Output

Gives a list of type *resbuf* containing the return code of the API.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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.

Registered Name

```
sproReleaseLicense
```

Glue Layer Wrapper

```
ADS_ReleaseLicense
```

Prototype

```
RB_WORD ADS_ReleaseLicense (char *address,
                            char *numSubLic);
```

Inputs

Name	Direction	Parameter Type	Description
address	IN	char*	The cell address of the sublicense. If a sublicense is to be released, specify the sublicense cell number, otherwise specify 0.
numSubLic	IN/OUT	char*	The number of sublicenses to be released. If the main license is to be released, this can be specified as zero or left blank.

Output

Gives a list of type *resbuf* containing the return code.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 partcular developer ID specified.

Registered Name

```
sproEnumServer
```

Glue Layer Wrapper

```
ADS_EnumServer
```

Prototype

```
RB_WORD ADS_EnumServer (char *enumFlag,
                        char *developerId,
                        char *serverInfo);
```

Inputs

Name	Direction	Parameter Type	Description
enumFlag	IN	char*	The flag used for contacting either: - the first-found server that has licenses to offer (NSPRO_RET_ON_FIRST_AVAILABLE), or - the first-found server that may have licenses (NSPRO_RET_ON_FIRST), or - all the SuperPro servers in the network (NSPRO_GET_ALL_SERVERS).

Name	Direction	Parameter Type	Description
developerId	IN	char*	The developer ID of the SuperPro key to find. Only the SuperPro servers running on the system having a key of matching developer ID will respond. If developer ID is specified as 0xFFFF then all the SuperPro servers (for a specified protocol) in the subnet will respond.
serverInfo	IN/OUT	char*	A pointer to a buffer that will contain the SuperPro server information, such as the computer address and the number of licenses available.

Output

Gives a list of type *resbuf* containing the return code of the API, the number of servers with their address, and number of licenses contained by each.

Return Code

On success returns AcRx::kRetOK, else AcRx::kRetError.

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 Type Definitions

- `typedef unsigned long  int* RBP_DWORD;`
- `typedef unsigned long  int  RB_DWORD;`
- `typedef int                 ENUM_SERVER_FLAG;`
- `typedef unsigned short int* RBP_WORD;`
- `typedef unsigned short int  RB_WORD;`
- `typedef unsigned short int  SP_STATUS;`
- `typedef unsigned short int  PROTOCOL_FLAG;`
- `typedef unsigned char*      RBP_BYTE;`
- `typedef unsigned char       RB_BYTE;`
- `typedef void*               RBP_VOID;`

Constants

- `#define SP_SUCCESS          0`
- `#define SP_API`
- `#define MAX_ADDR_LEN        32 /* Maximum host address length */`
- `#define MAX_NAME_LEN        64 /* Maximum host name length */`

Enumeration Flag Definition

```
/* Flags to specify the way of enumerating the SuperPro servers */

#define NSPRO_RET_ON_FIRST          1
#define NSPRO_GET_ALL_SERVERS       2
#define                             4
NSPRO_RET_ON_FIRST_AVAILABLE
```

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             __TEXT("RNBO_STANDALONE")
#define RNBO_SPN_DRIVER             __TEXT("RNBO_SPN_DRIVER")
#define RNBO_SPN_LOCAL              __TEXT("RNBO_SPN_LOCAL")
#define RNBO_SPN_BROADCAST          __TEXT("RNBO_SPN_BROADCAST")
#define RNBO_SPN_ALL_MODES          __TEXT("RNBO_SPN_ALL_MODES")
#define RNBO_SPN_SERVER_MODES       __TEXT("RNBO_SPN_SERVER_MODES")
```

Heartbeat Definition

```
/* To make the license update time programmable */

#define MAX_HEARTBEAT               2592000     /* 30*24*60*60 seconds */
#define MIN_HEARTBEAT               60          /* 60 seconds */
#define INFINITE_HEARTBEAT          0xFFFFFFFF  /* For infinite heartbeat */
```

Monitoring Information Structure Definition

```
/* Information about the key, used as a 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;
```

```
/* Provides information of the SuperPro server with 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;
```

Server Structure Definition

```
/* The SuperPro Server information with the number of licenses available */

typedef struct
{
   char      serverAddress[MAX_ADDR_LEN];
   RB_WORD   numLicAvail;
}NSPRO_SERVER_INFO;
```

Glue Layer

The AutoCAD interface makes use of a glue layer to call SuperPro APIs from the desired compiler to the Rainbow C-based library. The glue layer acts as an intermediary for passing data back and forth so that one can use an AutoCAD compiler and communicate to the standard C- based library.

Building the Glue Layer

This section elaborates on how to build the glue layer. However, you must fulfill the compiler requirements to build the glue layer.

Compilers Required

- Microsoft Visual C++ 6.0
- ObjectARX16 for AutoCAD 2000/ ObjectARX2002 for AutoCAD 2002

Input Files/Libraries

- *spromeps.h*
- *spromeps.lib*

Build Instructions

You can build the glue layer *sproARX.arx.* using the workspace in Microsoft Visual C++ IDE or using the batch file (*build.bat*).

Using Microsoft Visual C++ IDE

1. Open *sproeval.dsw* in Microsoft Visual C++ 6.0 IDE.
2. Check the path for the ARX library and include files in your project settings.
3. Click **Rebuild All** under the **Build** menu option.

Using *build.bat*

1. Check the path for the ARX library, include files and *vcvars32.bat* file manually in *build.bat*.
2. Type **`build sproARX`** at the command prompt to run the file.

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** – API Status Codes not listed below are obsolete even though they appear in the* spromeps.h *header file.*

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.
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.

Status Code (Decimal)	Description
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.
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.

Status Code (Decimal)	Description
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.
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.