

*Developer's  
Guide*

**SENTINEL** SuperPro

 **RAINBOW**  
TECHNOLOGIES

© Copyright 2000 – 2001, 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.

SentinelSuperPro is a trademark of Rainbow Technologies, Inc. Novell and NetWare are trademarks of Novell, Inc. Microsoft Windows, Microsoft Windows NT, Windows, Windows 95, Windows 98 and Windows 2000 are trademarks of Microsoft Corporation in the United States and other countries. All other product names referenced herein are trademarks or registered trademarks of their respective manufacturers.

#### CONFIDENTIAL INFORMATION

The SentinelSuperPro software protection system is designed to protect your software products from unauthorized use. The less information that unauthorized people have regarding your security system, the greater your protection. It is in your best interest to protect the information herein from access by unauthorized individuals. Please read the Developer's Agreement at the beginning of this guide for safeguarding requirements.

Part Number 700662-001, Revision B  
Software releases 6.1 and later

Revision	Action/Change	Date
A	Initial Release	October 2000
B	Fixed minor errors found in first revision	March 2001

#### **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 Strasse 7, D-82110 Germering, Germany  
Telephone: (49) 89 32 17 98 0 Fax: (49) 89 32 17 98 50

Additional offices are in the United States, Australia, China, India, the Netherlands, Russia and Taiwan. Distributors are located worldwide.



## Software License and Developer's Agreement

All Products (including developer's kits, Sentinel hardware keys, diskettes or other magnetic media, software, documentation and all future orders) are subject to the terms stated below. If you disagree with these terms, please return the Product and the documentation to Rainbow, postage prepaid, within three days of your receipt, and Rainbow will provide you with a refund, less freight and normal handling charges.

1. You may not copy or reproduce all or any part of the Product, except as authorized in item 2 below. Removal, emulation or reverse-engineering of all or any part of the Product constitutes an unauthorized modification to the Product and is specifically prohibited. Nothing in this license permits you to derive the source code of the software files that Rainbow has provided to you. Your software programs must be protected or licensed using a licensed and registered copy of this Rainbow Product. Rainbow provides no other warranty to any person, other than the Limited Warranty provided to the original purchaser of this Product.
2.
  - a. You may make archival copies of the software files and you may modify and merge them into your software programs for the sole purpose of implementing the Product to protect and/or license your programs according to the Rainbow documentation provided with the Product. All software files remain Rainbow's exclusive property.
  - b. Rainbow's Sentinel System Driver Software and other Rainbow software files listed in the "Licensee Redistribution Allowances" section (if it is defined in the Product's documentation) may be copied and distributed to your customers for the sole purpose of executing your protected or licensed software programs according to the Rainbow documentation provided with the Product.
  - c. No license is granted to Licensee to sell, license, distribute, market or otherwise dispose of any software files or other component of the Product except when embedded in your software programs. Copies of your software programs must bear a valid copyright notice and must be distributed such that the object code for the Product cannot be extracted.
3. Rainbow warrants the Product and the magnetic media on which the software files are provided to be substantially free from significant defects in materials and workmanship under normal use for a period of twelve (12) months from the date of delivery of the Product to you. In the event of a claim under this warranty, Rainbow's sole obligation is to replace or repair, at Rainbow's option, any Product free of charge. Any replaced parts shall become Rainbow's property.
4. Warranty claims must be made in writing during the warranty period and within seven (7) days of the observation of the defect, accompanied by evidence satisfactory to Rainbow. Prior to returning any

Product to Rainbow, you must obtain a Return Material Authorization (RMA) number and shipping instructions from Rainbow. Products returned to Rainbow shall be shipped with freight and insurance paid.

5. Except as stated above, there is NO OTHER WARRANTY, REPRESENTATION, OR CONDITION REGARDING RAINBOW'S PRODUCTS, SERVICES, OR PERFORMANCE, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Rainbow is not responsible for any delays beyond its control. Rainbow's entire liability for damages to you or any other party for any cause whatsoever, whether in contract or in tort, including negligence, shall not exceed the price you paid for the unit of Product that caused the damages or that are the subject matter of, or are directly related to, the cause of action. In no event will Rainbow be liable for any damages caused by your failure to perform your obligations, or for any loss of data, profits, savings, or any other consequential and incidental damages, or for any claims by you based on any third-party claim.

## ***Licensee Redistribution Allowances***

SentinelSuperPro Licensees may release the Sentinel System Driver (sentinel.sys, sentinel.vxd) and the Sentinel Client Activator, as well as any associated files for installation with their SentinelSuperPro-protected application. In addition, the Licensee may distribute the following commands, files and related documentation: spcom-mon.dll, sp\_g24.dll, sp\_g08.dll, sp\_g04.dll, makedll.dll, dsafedll.dll, dsafe32.dll, usafe32.dll, lang\_enu.dll, fieldexutil.exe, fieldexutil.chm, licensegenutil.exe, licensegenutil.chm, makekeysutil.exe, makekeysutil.chm, instdrv.exe, instdrv.c, sentdata.vxd, monitor.exe, ssp\_6\_1\_monitoring\_tool.chm, loadserv.exe, spnsrvnt.exe, spnsrv9x.exe, the *SentinelSuperPro System Administrator's Guide* and Chapter 14 of the *SentinelSuperPro Developer's Guide*.

## **FCC Notice to Users**

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. Operation is subject to the following conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment generates, uses, and can radiate frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If interference problems do occur, please consult the system equipment's owner's manual for suggestions. Some of these suggestions include relocation of the computer system away from the television or radio, or placing the computer AC power connection on a different circuit or outlet.

Change or modifications to this product without the express approval of Rainbow Technologies, Inc., could result in non-FCC compliance, and void the user's authority to operate this equipment.

## International Quality Standard Certification



Certificate Number FM 30128

Rainbow Technologies, Inc. Irvine, CA facility has been issued the ISO 9002 Certification, the globally recognized standard for quality, by British Standards Institution as of December 1994.

## European Community Directive Conformance Statement



This product is in conformity with the protection requirements of EC Council Directive 89/336/EEC. Conformity is declared to the following applicable standards for electro-magnetic compatibility immunity and susceptibility; CISPR22 and IEC801. This product satisfies the CLASS B limits of EN 55022.



# Contents

---

## Contents

Conventions Used in This Guide .....	xvii
How to Get the Most from This Guide.....	xviii
Accessing Online Documentation .....	xx
<i>Using Online Help</i> .....	xx
Accessing SentinelSuperPro Documentation.....	xxi
Getting Help.....	xxiii
We Welcome Your Comments .....	xxv

## Chapter 1 – What Is SentinelSuperPro?

SentinelSuperPro Components .....	2
<i>The Hardware Key</i> .....	2
<i>The SentinelSuperPro API</i> .....	4
<i>The SentinelSuperPro Developer’s Toolkit</i> .....	4
<i>The SentinelSuperPro Server</i> .....	5
<i>The SentinelSuperPro Monitoring Tool</i> .....	5
How SentinelSuperPro Protects Your Software .....	6
<i>Protection Types</i> .....	8
SentinelSuperPro Features and Benefits .....	10
<i>What’s New in SentinelSuperPro 6.1?</i> .....	14
What’s Included with SentinelSuperPro 6.1? .....	15
System Requirements.....	17
<i>Minimum Hardware Requirements</i> .....	17
<i>Minimum Software Requirements</i> .....	17

**Chapter 2 – Installation**

Running SentinelSuperPro Setup ..... 20

*Preparing to Install* ..... 20

*Installing SentinelSuperPro Components* ..... 22

Installing the SentinelSuperPro Hardware Key ..... 27

*Installing the Parallel Port Hardware Key* ..... 28

*Installing the USB Hardware Key* ..... 30

Installing SentinelSuperPro Interfaces ..... 31

Installing the Sentinel Client Activator ..... 32

**Chapter 3 – Using the Hardware Key**

Getting to Know the Key ..... 34

*Restricted Cells* ..... 35

*Programmable Cells* ..... 36

Programming the Key ..... 36

*Access Codes* ..... 36

*Cell Types* ..... 37

*Cell Values* ..... 45

*Valid Algorithm Addresses* ..... 47

Ordering and Returning Keys ..... 52

*Ordering Additional Keys* ..... 52

*Returning Keys* ..... 53

**Chapter 4 – Designing Your Protection Strategy**

Introduction to Software Security Concepts ..... 57

*Protection Types* ..... 57

*Active v. Inactive Applications* ..... 59

*Activation Types* ..... 60

*Network Licenses* ..... 68

Getting Started ..... 69

*Quick and Easy Protection* ..... 69

*Customized Protection* ..... 69

*Basic Protection Guidelines* ..... 70



Commonly Used Protection Techniques .....	72
<i>Reading Stored Data</i> .....	72
<i>Using Algorithms to Encrypt Data</i> .....	74
<i>Using Activation Passwords</i> .....	76
<i>Dealing With Missing Hardware Keys</i> .....	78
<i>Dealing With Newly Connected Hardware Keys</i> .....	80
Advanced Protection Techniques .....	81
<i>Using Returned Values as Variables</i> .....	81
<i>Implementing Encryption Techniques</i> .....	81
<i>Querying Activation Passwords</i> .....	87
<i>Using Data Words</i> .....	87
<i>Assembly Language Techniques</i> .....	88
<i>Using Stepped Access</i> .....	89
<i>Obstructing Debuggers</i> .....	90
Controlling Demo Applications .....	90
<i>Using Counters</i> .....	91
<i>Querying Counters</i> .....	95
Programming the Hardware Key .....	95
<i>Using One Key for Multiple Applications</i> .....	96
Moving On .....	97

## Chapter 5 – Implementing Licensing

Setting the Access Mode .....	100
<i>Setting Stand-alone or Network Mode</i> .....	100
<i>Setting Dual Mode</i> .....	101
<i>About the NSP_HOST Variable</i> .....	102
Finding a Key .....	103
<i>Finding a Key in Stand-alone Mode</i> .....	103
<i>Finding a Key in Network Mode</i> .....	103
<i>Finding a Key in Dual Mode</i> .....	104
Getting a License .....	105
<i>The License ID</i> .....	106
<i>Maintaining the License</i> .....	106
<i>A Note About Licenses</i> .....	106
Releasing a License .....	107

Using Sublicenses .....	107
<i>Sublicense Usage Example</i> .....	108
<i>Getting a Sublicense</i> .....	108
<i>Adding Sublicenses to Your Protection Strategy</i> .....	109
 <b>Chapter 6 – Starting the SentinelSuperPro Toolkit</b>	
Opening the SentinelSuperPro Toolkit .....	112
<i>Entering Your Passwords</i> .....	112
<i>Including Overwrite Passwords in the Field Exchange DLLs</i> .....	115
<i>Enabling the One-Time Update Option for License Codes</i> .....	115
Navigating in the SentinelSuperPro Toolkit .....	117
<i>Stages and Sections</i> .....	117
<i>Menu Bar</i> .....	122
<i>Getting Help</i> .....	122
<i>Using Online Help</i> .....	123
Completing the Overview Stage .....	123
<i>Learning About SentinelSuperPro Concepts</i> .....	123
<i>Using the API Explorer</i> .....	123
<i>Querying Algorithms</i> .....	129
Creating a Project .....	131
<i>What Is a Project?</i> .....	131
<i>Creating a New Project</i> .....	132
<i>Importing a .DAT File</i> .....	132
<i>Changing Your Developer ID or Passwords</i> .....	133
Opening an Existing Project .....	134
Saving Your Project .....	135
Adding Password Protection to Your Project .....	135
<i>Locking a Project</i> .....	136
<i>Changing the Password for a Locked Project</i> .....	137
<i>Unlocking a Project</i> .....	137
Creating a Project File for Distributors .....	138
Closing the SentinelSuperPro Toolkit .....	139

## Chapter 7 – Protecting Your Application

What Is Application Protection? .....	142
<i>Demo Applications</i> .....	143
Selecting a Protection Type .....	145
Using Integrated Protection .....	146
<i>Selecting the First Cell</i> .....	147
<i>Adding a Demo Counter</i> .....	147
<i>Overriding the Default Algorithm Values</i> .....	148
<i>Selecting the Activation Type</i> .....	149
Using Automatic Protection.....	152
<i>Selecting the First Cell</i> .....	152
<i>Overriding the Default Algorithm Values</i> .....	153
<i>Selecting the Input and Output Files</i> .....	154
<i>Selecting Automatic Protection Demo Options</i> .....	156
<i>Selecting Additional Files for Encryption</i> .....	160
<i>Selecting the Activation Type</i> .....	163
<i>Customizing Error Messages</i> .....	165
Protecting Multiple Applications.....	166
Editing an Application Protection Element .....	166
<i>Deleting an Application Protection Element</i> .....	167
Where to Go from Here.....	167

## Chapter 8 – Working With Design Elements

Custom Element Types .....	170
Adding Algorithms.....	172
<i>Entering Counter Values</i> .....	175
<i>Entering Password Values</i> .....	176
Adding Counters .....	177
Adding Data Words .....	179
Adding Sublicense Limits .....	182
Editing Existing Elements .....	184
<i>Deleting an Element</i> .....	184
Rearranging Elements on the Key .....	185

**Chapter 9 – Implementing Your Strategy**

Creating the Prototype ..... 188

*Starting the Prototype Process*..... 189

Verifying the Key Using MemView ..... 190

Adding API Functions to Your Source Code ..... 192

*Viewing the Pseudocode*..... 192

*Adding Code to Your Application*..... 194

*Using the API Explorer to Evaluate the API Functions*..... 194

Shelling an Application ..... 196

Testing Your Application Protection..... 198

**Chapter 10 – Defining Field Activation Actions**

Working with Actions ..... 202

*Adding an Action*..... 203

*Removing an Action* ..... 204

Working with Commands ..... 205

*Adding a Command* ..... 205

*Removing a Command*..... 208

*Available Commands* ..... 209

Testing Your Strategy ..... 210

Final Steps ..... 211

**Chapter 11 – Programming Keys**

Setting Up to Program Product Keys ..... 215

*Selecting the Appropriate Keys*..... 215

*Connecting the Keys* ..... 215

Programming a Product Key ..... 217

*Viewing Programming Statistics*..... 219

*Verifying the Key Was Programmed Correctly* ..... 220

Setting Up to Program Distributor Keys ..... 221

*Selecting the Appropriate Keys*..... 221

*Connecting the Keys* ..... 221

Programming a Distributor Key ..... 223

## Chapter 12 – Shipping Your Application

What to Send to Your Customers .....	228
<i>Installing the Sentinel System Driver</i> .....	229
<i>Installing the SentinelSuperPro Server</i> .....	230
<i>Installing the Sentinel Data Protection Driver</i> .....	236
What to Send to Your Distributors .....	238
<i>Customer Items</i> .....	238
<i>Distributor-Only Items</i> .....	239
Packaging and Handling Guidelines for Keys .....	241

## Chapter 13 – Activating and Updating Keys

How Product Keys are Activated or Updated .....	244
<i>What Is a Locking Code?</i> .....	245
<i>What Is a License Code?</i> .....	246
How Distributors Activate an Application .....	247
Using the Client Activator .....	249
<i>Client Activator Customer Requirements</i> .....	250
<i>Steps for Deploying the Client Activator</i> .....	250
<i>If You Don't Use the Client Activator</i> .....	251
Updating Product Keys in the Field .....	252
<i>Receiving the Locking Code from Your Customer</i> .....	252
<i>Generating a License Code</i> .....	252
<i>Sending the License Code to Your Customer</i> .....	255
Updating Distributor Keys in the Field .....	257

## Chapter 14 – Using the Stand-alone Utilities

Verifying the SentinelSuperPro Server Is Running .....	261
Using the Make Keys Utility .....	262
<i>Installing the Make Keys Utility</i> .....	262
<i>Programming Product Keys</i> .....	264
<i>Programming Distributor Keys</i> .....	266
<i>Viewing Programming Statistics</i> .....	268
Using the License Generator Utility .....	269
<i>Installing the License Generator Utility</i> .....	269
<i>Opening the License Generator Utility</i> .....	270
<i>Generating a License Code</i> .....	271

Contents

Using the Field Exchange Utility..... 274

*Installing the Field Exchange Utility – Developers* ..... 274

*Installing the Field Exchange Utility – Customers* ..... 274

*Opening the Field Exchange Utility* ..... 276

*Generating a Locking Code* ..... 277

*Entering a License Code* ..... 278

**Chapter 15 – API Function Reference**

Using the SentinelSuperPro API ..... 282

API Functions Summary ..... 284

RNBOSproActivate ..... 286

RNBOSproDecrement ..... 288

RNBOSproEnumServer ..... 290

RNBOSproExtendedRead ..... 292

RNBOSproFindFirstUnit..... 293

RNBOSproFindNextUnit ..... 294

RNBOSproFormatPacket ..... 295

RNBOSproGetContactServer ..... 296

RNBOSproGetFullStatus ..... 297

RNBOSproGetHardLimit ..... 298

RNBOSproGetKeyInfo ..... 299

RNBOSproGetSubLicense ..... 301

RNBOSproGetVersion ..... 302

RNBOSproInitialize ..... 304

RNBOSproOverwrite ..... 305

RNBOSproQuery ..... 307

RNBOSproRead..... 310

RNBOSproReleaseLicense ..... 311

RNBOSproSetContactServer ..... 312

RNBOSproWrite..... 313

API Status Codes ..... 315

**Chapter 16 – Migrating From SentinelSuperPro 5.1 or NetSentinel**

Finding 5.1 Features in SentinelSuperPro 6.1 ..... 322

Finding NetSentinel Features in SentinelSuperPro 6.1 ..... 325

## **Appendix A – Using the Command Line Shell Utility**

Command Line Syntax .....	328
Using the Shell Utility .....	329
<i>Example</i> .....	329

## **Appendix B – Troubleshooting**

Uninstalling the SentinelSuperPro Toolkit.....	332
Repairing a SentinelSuperPro Installation.....	334
Strategy Design Issues .....	336
Application Protection Issues .....	337
<i>Protecting Multi-File Applications</i> .....	338
<i>Protecting Interpreted-language Applications</i> .....	339
<i>Input File Attributes</i> .....	340
<i>Thread Local Storage</i> .....	340
<i>Lahey F90 Fortran 2.0</i> .....	340
<i>Protecting FoxPro 3.0 and 5.0 Applications</i> .....	340
<i>Protecting Microsoft J++ 1.1 Java Applets</i> .....	340
<i>Note for SmartHeap Users</i> .....	341
<i>Protecting Applications That Use “Starter” Programs</i> .....	341
Key Programming Issues.....	342
Application Activation Issues.....	344
SentinelSuperPro Compatibility .....	346
SentinelSuperPro Key Compatibility Issues .....	347
<i>About the Parallel Hardware Interface</i> .....	347
<i>About the USB Hardware Interface</i> .....	349

## **Appendix C – Compatible Compilers and Applications ..... 351**

## **Appendix D – Glossary ..... 355**

## **Index ..... 371**





# Preface

---

Thank you for selecting SentinelSuperPro to protect your applications from unauthorized use. The SentinelSuperPro software protection system combines a programmable hardware key with the ability to encrypt data, giving you a wide range of methods for securing up to 28 applications per key from illegal distribution and use.

---

## Conventions Used in This Guide

Please note the following conventions concerning bold lettering, italics and more:

Convention	Meaning
Select	Use the arrow keys or mouse to select an item on a menu, a field in a window or an item in a list.
Click	Press the primary mouse button once. The primary mouse button is typically the left button, but may be reassigned to the right button.
Courier	Denotes syntax, prompts and code examples. If bold, denotes text you type.
<b>Bold Lettering</b>	In procedures, words in boldface type represent keystrokes, menu items, window names or mouse commands.
<i>Italic Lettering</i>	Words in italics represent file names and directories, or, when used in explanatory text, for emphasis.

# How to Get the Most from This Guide

The *SentinelSuperPro 6.1 Developer's Guide* walks you through the entire process of protecting your applications, including planning, protecting, packaging and shipping a protected application to your customers. The following table explains what you can find in each chapter of this guide:

Chapter/Appendix	Description
Chapter 1 – What Is SentinelSuperPro?	An overview of SentinelSuperPro components, features and benefits, including system requirements and what's new in 6.1.
Chapter 2 – Installation	Instructions for installing the Toolkit, the network server, the monitoring tool, the hardware key and the Sentinel driver.
Chapter 3 – Using the Hardware Key	An introduction to the SentinelSuperPro hardware key, with descriptions of the physical key layout, memory cells and algorithm values and addresses.
Chapter 4 – Designing Your Protection Strategy	Information about techniques you can use in your protection strategy, and an explanation of the basics of software protection with SentinelSuperPro.
Chapter 5 – Implementing Licensing	Describes how to use network licenses with your protected application, including instructions for obtaining, maintaining and releasing licenses across a network.
Chapter 6 – Starting the SentinelSuperPro Toolkit	Instructions for opening and navigating in the Toolkit, using the API Explorer and creating and saving project files.
Chapter 7 – Protecting Your Application	Describes how to apply integrated or automatic (shelled) application protection to your applications.
Chapter 8 – Working With Design Elements	Procedures for adding custom elements—algorithms, counters, data words and sublicenses—to your protection strategy.

Chapter/Appendix	Description
Chapter 9 – Implementing Your Strategy	Information on creating a prototype key, adding a shell to an application and adding API function calls to your source code.
Chapter 10 – Defining Field Activation Actions	Information about defining actions and commands used in field activation.
Chapter 11 – Programming Keys	Describes how to program product keys and distributor keys to ship with your protected application.
Chapter 12 – Shipping Your Application	Provides lists of the items you need to send along with your application to both customers and distributors.
Chapter 13 – Activating and Updating Keys	Explains how to retrieve information about a key from your customer or distributor, and how to generate a code to update product or distributor keys.
Chapter 14 – Using the Stand-alone Utilities	Instructions for using the three SentinelSuperPro stand-alone utilities: the License Generator Utility, the Field Exchange Utility and the Make Keys Utility.
Chapter 15 – API Function Reference	An overview of SentinelSuperPro API functions for use with Win-32 applications, including parameters, return values and status codes.
Chapter 16 – Migrating From SentinelSuperPro 5.1 or NetSentinel	Assists you with finding where features used in SentinelSuperPro 5.1 or NetSentinel 5.31 are located in SentinelSuperPro 6.1.
Appendix A – Using the Command Line Shell Utility	Describes how to use the included command-line version of automatic (shelled) protection to shell applications during the development phase of your product.
Appendix B – Troubleshooting	Presents common problems you may encounter while using SentinelSuperPro and solutions to those problems.

Chapter/Appendix	Description
Appendix C – Compatible Compilers and Applications	A list of the compatible compilers and applications that can be used to develop applications to be protected with SentinelSuperPro 6.1.
Appendix D – Glossary	A glossary of SentinelSuperPro and software protection terms used throughout this manual.

---

# Accessing Online Documentation

There are several ways to get help while using the SentinelSuperPro Developer’s Toolkit. For general issues, look for answers in this guide and in the online Help system that is included with the Toolkit.

You may also want to read through the text provided in the Toolkit’s Overview stage. The introductory information included there can help you gain a basic understanding of SentinelSuperPro concepts.

Additionally, as you move through the stages, pay attention to the text that appears in the *orientation pane* at the top of the Toolkit window. This text provides a quick overview of the steps you’ll take in each stage and how they apply to protection strategies. If you find yourself unsure of what to do in a particular stage, read the orientation pane text for help.


## Using Online Help

The SentinelSuperPro Developer’s Toolkit ships with a complete online Help system. It includes a detailed table of contents and thorough index searching capabilities.

Toolkit Help is very easy to use. The majority of the information found in this guide is also available through Help.

To access online Help, select **Help Topics** from the **Help** menu.

Most fields also have context-sensitive help associated with them, which is accessible in any of three ways:

- Right-click on a field, check box or button to view Help information specific to that item.
- Click in a field or on a check box, and then press **F1**.
- Click the **What's This** button  at the bottom of the Toolkit window to access the Help pointer, then click on the item you need help for.

For more information, please review the “Using Help” topic in online Help.

## Accessing SentinelSuperPro Documentation

If you can't find the answer you need in Help, refer to the SentinelSuperPro documentation for more detailed information and instructions.

Manual	What's In It?	Who Should Read It?
<i>SentinelSuperPro 6.1 Quick Start Guide</i>	A quick tour of SentinelSuperPro for Windows application developers.	Anyone who is new to SentinelSuperPro or software protection and wants a quick overview of SentinelSuperPro features.
<i>SentinelSuperPro 6.1 System Administrator's Guide</i>	Instructions for installing and running the SentinelSuperPro Server and Monitoring Tool.	Developers who will be implementing network functionality, and system administrators responsible for deploying a protected application in their organization.
<i>SentinelSuperPro 6.1 Developer's Guide</i>	All the steps necessary to protect, package and ship applications protected with SentinelSuperPro, as well as information about the SuperPro API calls.	Developers responsible for the overall process of protecting and shipping a Windows application.

Each of these guides are available in portable document format (PDF), and are installed on your computer during SentinelSuperPro setup.

You need Adobe Acrobat Reader to view and print PDF files. We recommend installing Acrobat Reader 4.0 or higher for best results. This version of Acrobat can be installed from the SentinelSuperPro CD.

Once you have installed Acrobat Reader, you are ready to access the documentation PDF files. To do so, navigate to **Start > Programs > Rainbow Technologies > SuperPro > 6.1**.

---

**Tip:** Check the Rainbow Technologies Web site ([www.rainbow.com](http://www.rainbow.com)) for the most up-to-date information about SentinelSuperPro, including FAQs, technical notes and the latest versions of SentinelSuperPro documentation.

---

## Getting Help

Rainbow Technologies is committed to supporting SentinelSuperPro. 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

<b>Corporate Headquarters North America and South America</b>	
	Rainbow Technologies North America
Internet	<a href="http://www.rainbow.com/support.html">http://www.rainbow.com/support.html</a>
E-mail	<a href="mailto:techsupport@rainbow.com">techsupport@rainbow.com</a>
Telephone	(800) 959-9954 (6:00 a.m. – 6:00 p.m. PST)
Fax	(949) 753-9510
<b>Australia</b>	
	Rainbow Technologies (Australia) Pty Ltd.
E-mail	<a href="mailto:techsupport@au.rainbow.com">techsupport@au.rainbow.com</a>
Telephone	(61) 3 9820 8900
Fax	(61) 3 9820 8711
<b>China</b>	
	Rainbow Information Technologies (China) Co.
E-mail	<a href="mailto:Sentinel@isecurity.com.cn">Sentinel@isecurity.com.cn</a>
Telephone	(86) 10 8266 3936
Fax	(86) 10 8266 3948
<b>France and Distributors in Europe, Middle East and Africa</b>	
	Rainbow Technologies
E-mail	<a href="mailto:techsupport@fr.rainbow.com">techsupport@fr.rainbow.com</a>
Telephone	(33) 1 41.43.29.00
Fax	(33) 1 46.24.76.91

Rainbow Technologies Technical Support Contact Information (Continued)

Germany	
	Rainbow Technologies, GmbH
E-mail	techsupport@de.rainbow.com
Telephone	(49) 89 32 17 98 0
Fax	(49) 89 32 17 98 50
Taiwan	
	Rainbow Technologies (Taiwan) Co.
E-mail	techsupport@tw.rainbow.com
Telephone	(886) 2 2570-5522
Fax	(886) 2 2570-1988
United Kingdom and Ireland	
	Rainbow Technologies, Ltd.
E-mail	techsupport@uk.rainbow.com
Telephone	(44) 1932 579200
Fax	(44) 1932 570743



---

## We Welcome Your Comments

To help us improve future versions of SentinelSuperPro documentation, we want to know about any corrections, clarifications or further information you would find useful. When you contact us, please include the following information:

- The title and version of the guide you are referring to
- The version of SentinelSuperPro you are using
- Your name, company name, job title, phone number and e-mail address

Send us e-mail at:

**techpubs@rainbow.com**

Or, you can write us at:

**Rainbow Technologies, Inc.  
50 Technology Drive  
Irvine, CA 92618**

**Attn: Technical Publications Department**

Thank you for your feedback.



# Chapter 1

---

## What Is SentinelSuperPro?

In addition to providing you with a full-featured, easy-to-use software protection system, SentinelSuperPro gives you the ability to increase demo limits, upgrade demos to fully-licensed versions and provide access to additional features all without having to ship a new hardware key or visit the customer's site.

SentinelSuperPro 6.1 provides you with an added capability—the ability to allow your customers to use one key for multiple clients. SentinelSuperPro 6.1 also allows you to program keys specifically for use by your distributors, so you can limit how many product keys they can activate and update.

This chapter covers the following topics:

- SentinelSuperPro 6.1 components
- How SentinelSuperPro protects your software
- SentinelSuperPro 6.1 features and benefits
- What's new in SentinelSuperPro 6.1
- What's included with SentinelSuperPro 6.1
- System requirements for using SentinelSuperPro 6.1

## SentinelSuperPro Components

The SentinelSuperPro system is made up of five components:

- The hardware key
- The SentinelSuperPro API
- The SentinelSuperPro Developer's Toolkit
- The SentinelSuperPro Server
- The SentinelSuperPro Monitoring Tool

Each of these components is explained in the following sections.

### The Hardware Key

The SentinelSuperPro hardware key is a programmable, read/write memory device that provides the responses necessary to unlock your application. The hardware key is the heart of your application protection strategy.

To implement a protection scheme, you program your application to send calls to the hardware key to verify its presence. If the correct hardware key is attached to the user's system or available on the network, it responds to your application's calls with the appropriate responses, allowing the user access to your application.

Each key contains 64 memory cells, 56 of which are available for programming by you. These memory cells can be programmed with algorithms, data values to provide fixed responses, or to serve as counters. Each key also contains internal logic that transforms data based on encryption strings you define.

## ***Network Keys v. Stand-alone Keys***

There are two types of SentinelSuperPro hardware keys: *network* and *stand-alone*:

- The *network key* allows multiple network clients to access a protected application using a single hardware key. Network keys, which are typically connected to servers on the network (see page 5), are programmed at the factory with a *hard limit*.

The hard limit defines the maximum number of licenses that can be obtained from the key, and thus the maximum number of users (both local and across the network) that can access the protected application. Keys are available with the following pre-programmed hard limits: 1, 2, 3, 5, 10, 25, 50 or unlimited.

- A *stand-alone key* is typically connected directly to a user's local workstation, providing access to the protected application only on a single system. Stand-alone keys have a hard limit of 0, meaning the key can be used by any number of local users. These keys can also be connected to servers, but provide only a single license at any one time.

## ***Product Keys v. Distributor Keys***

Prior to shipping your application to your customers, you must program your hardware keys with your protection strategy. A hardware key can be programmed as either a *product key* or a *distributor key*.

- *Product keys* are shipped to your end users with your protected application, providing access to the application. Product keys may be either stand-alone or network keys, depending on how your application will be used (by single clients or across the network).
- *Distributor keys* are given to your sales distributors, allowing them to perform activation and update functions on product keys provided to end users when they sell your protected application. Distributor keys can be either stand-alone or network keys; they must be connected to the distributor's local machine.

Rainbow Technologies customizes SentinelSuperPro hardware keys for each developer, which means another developer cannot reprogram your keys.

---

**Tip:** Refer to Chapter 11, “Programming Keys,” on page 213 for more information about programming product and distributor keys. For more information about how keys are activated and updated, refer to Chapter 13, “Activating and Updating Keys,” on page 243.

---

### The SentinelSuperPro API

The SentinelSuperPro API is a set of functions used to communicate between your application, the Sentinel system driver, the server and the hardware key. If you choose to use the *integrated* protection option (see page 8), you embed API function calls to communicate with the hardware key directly in your application’s source code.

### The SentinelSuperPro Developer’s Toolkit

The SentinelSuperPro Developer’s Toolkit (SSP Toolkit) is a Windows application that combines the functions necessary to develop your protection strategy, program the hardware keys, and ship a protected application into one, easy-to-use package.

Once you have developed and prototyped your protection strategy using the SSP Toolkit, a protection plan with pseudocode is generated for you to use as a guide for adding the appropriate API function calls to your source code.

After you have modified your source code, or *shelled* your application (see page 9), you are ready to use the SSP Toolkit to program your hardware keys with the values your application will use to determine whether or not the key is attached to the user’s system or the server.

The following SuperPro utilities from previous versions of SentinelSuperPro are now combined into the Developer’s Toolkit:

- SentinelWizard
- SentinelShell

- SentinelSuperPro Advanced Editor
- SentinelSuperPro SAFE
- SentinelSuperPro Manufacturing Utility (SMU)
- Sentinel Evaluation Program
- Sentinel Query Response Generator

---

**Tip:** For more information about updating from SentinelSuperPro 5.1 to SentinelSuperPro 6.1, please refer to Chapter 16, “Migrating From SentinelSuperPro 5.1 or NetSentinel,” on page 321.

---

## The SentinelSuperPro Server

If you design your protected application to be run on a network using concurrent licensing, your customer must install the SentinelSuperPro Server on the same machine where the hardware key is located. This server manages licensing and security for the protected application. The server is the link between the client running your application and the hardware key that responds to the API functions used in your protection strategy.

## The SentinelSuperPro Monitoring Tool

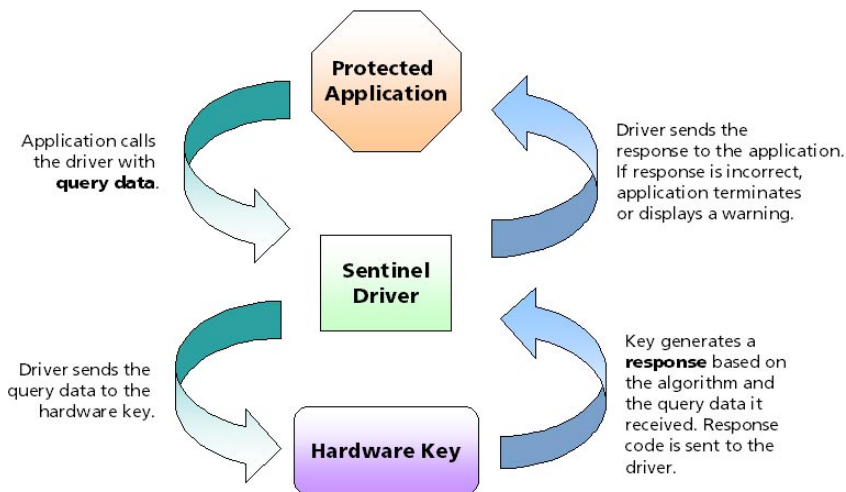
The SentinelSuperPro Monitoring Tool is a Windows application designed for use with protected applications intended to be run on a network. The Monitoring Tool displays information about all SentinelSuperPro servers, keys and user licenses in the field. The tool reports statistics, such as the number of licenses currently in use and the license limit for each key. Like the server, this application must be shipped to your customer with your protected application.

## How SentinelSuperPro Protects Your Software

At its most basic, SentinelSuperPro protects your software through a series of steps known as a *software lock*. Each software lock is a call to an API function that verifies the presence of the hardware key to succeed.

1. Your application calls the Sentinel driver, which communicates with the hardware key attached to an external port on the user's computer, sending a query string to an algorithm.
2. The key returns a response to the driver, which communicates back to your application.
3. Your application evaluates the response and acts accordingly.

An invalid response indicates the correct key is *not* attached or has been tampered with. Your application then terminates or displays a warning message. Software can be illegally copied, but it will not run.



How the Key Handles Application Calls



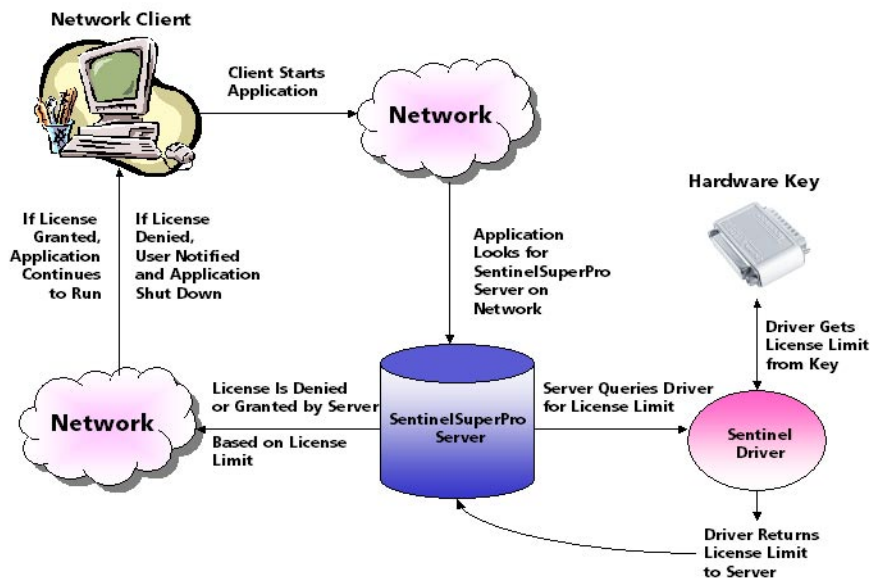
When a SentinelSuperPro-protected application is used on a network, software locks (see page 6) are performed across the network only after a license has been obtained.

1. Your application sends a call to find a hardware key and obtain a license. If the key is found on the user's local system, software locks are performed as explained on page 6.
2. If a key is *not* found on the user's system, the application sends a broadcast message to the network to locate a SentinelSuperPro server.
3. Once found, the server queries the Sentinel driver to obtain the license limit from a hardware key attached to an external port on the server.
4. The driver reads the license limit in the key and returns it back to the server.
5. The server decides whether or not to grant the license and then sends the license information to your application.
6. After obtaining the license, your application sends periodic "heart-beat" messages to maintain the license. Failure to send a heartbeat message releases the license and returns an error to the application.
7. Software locks are performed as required by your application, using the license as permission to communicate with the key.
8. When all software locks are complete, the application releases the license back to the key through the server, allowing the license to be obtained by another client.

---

**Note:** The above procedure assumes use of the default access mode: **dual mode**. You can change how SentinelSuperPro obtains a license by changing the access mode. See "Setting the Access Mode" on page 100 for more information.

---



### How an Application Obtains a License over the Network

## Protection Types

SentinelSuperPro offers you two methods for protecting your application: *integrated* or *automatic*. When and where the software locks are implemented depends on the type of protection being used.

- **Integrated:** Integrated protection consists of software locks (API function calls) added directly to your source code. It is used to create a custom protection strategy, with control over the amount and location of software locks.

The frequency of software locks within your application, and the action taken if no key is found, is left up to you. The more locks you add to your application, the more difficult it will be for potential hackers to break your application's protection.

Because you must understand the API function calls used to support the protection strategy you have designed, and manually add them to your code, using integrated protection may take longer.

- **Automatic:** Automatic (*shelled*) protection is the fastest and easiest method of protecting your applications with SentinelSuperPro.

Instead of adding software locks to your source code, a protective “shell” is automatically added to your application’s executable file, so that the software lock is called before the application starts—if the hardware key is not present, the user sees an error message and the application does not run. Also, while the application is running, the shell periodically checks to verify the hardware key is still attached—if at any time the key is missing the application shuts down immediately.

Automatic protection also gives you more control over demo options such as expiration dates, counters and time/date limits.

## SentinelSuperPro Features and Benefits

- **Customizable Protection**

One key can be programmed to provide several different types of both fixed and variable responses, giving you many variations in the types of software locks you can create.

For example, cells can be used to store fixed user data, such as serial numbers, user names or codes controlling feature access. Such data can be read by your application to verify the key is still attached or to perform some other function. You can also use stored data to control program flow or application functions.

Cells can also store algorithms used to scramble query codes sent by your application. Other cells can be programmed as counters used to restrict the number of executions. While the first eight cells are reserved for system information, the other 56 cells in each key can be used any way you desire (with some restrictions).

- **Password Protection**

The ability to program SentinelSuperPro hardware keys is protected by three passwords: the *write* password and two *overwrite* passwords. The write password allows you to write to undefined cells and read/write data words. The two overwrite passwords allow you to write to all other non-restricted cells: read-only data words, counters and algorithm words.

You must have your passwords to program keys through the SSP Toolkit or the Make Keys Utility. You also must include the passwords in your protected application to reprogram cells in the field or use some API function calls. Passwords ensure only authorized users can change your protection strategy or program keys.

- **Field Exchange Capability**

Shipping your protected application and its corresponding key(s) to customers in the field doesn't end your control over the key and your software. With SentinelSuperPro, you can perform a number of functions on keys already in the field, including activating and updating product, setting or clearing bits, and incrementing or decrementing counters.

Field exchange enables you to ship your application in an unusable state, and provide a means for legitimate users to activate it. The activation process is protected by encryption algorithms and passwords pre-programmed into the key. This same process also allows you to support field upgrades and control feature access.

- **Demo Application Control**

If you provide demo or trial versions of your applications to your customers, you may want those applications to run only a set number of times, or you may want to define an expiration date. SentinelSuperPro gives you demo application control through the use of counters, time limits and expiration dates.

- **Multiple Applications Per Key**

With SentinelSuperPro, you can protect up to 28 applications on a single hardware key. In each protection strategy, certain cells in the key are assigned to each application. Each application can then query the key using algorithms. Thus, your users can run several protected applications with a single hardware key attached.

The number of applications you can assign to a single key is dependent on how complex your protection strategy is. More complicated strategies require more memory cells, resulting in fewer cells available for other protected applications.

- **License Enforcement**

A significant addition to SentinelSuperPro 6.1 is its ability to enforce concurrent licensing agreements. The SentinelSuperPro server keeps track of the number of licenses in use for each key, and each sublicensed application on the key, and does not grant new license requests once a key's license limit has been reached. When licenses are returned to the server by an application, they become available for reuse by another client.

The number of available licenses is determined by the hard limit programmed into the key, or through the use of sublicensing per application.

- **Sublicensing**

Sublicensing is useful when you want to apply a license limit that is less than the key's factory-programmed (*hard*) limit.

You can program up to 56 separate sublicense license limits in each key—each sublicense is a custom element occupying a single cell on the hardware key. The total number of sublicense limits you can program is dependent on the number of cells being used by other elements of your strategy.

In addition to defining your own license limits for the application as a whole, you can also use sublicenses to control concurrent access to specific features within a protected application. At runtime, access to a controlled feature is granted if the corresponding sublicense limit has not been exceeded.

- **Local or Network Access**

Using the SentinelSuperPro API, you can configure your application to run on a non-networked (*stand-alone*) system with a key directly attached, on a network using a license obtained from a key attached to a server, or on either a stand-alone system or a network, depending on how the application is being used.

- **Multiple Key and Server Support**

Up to 10 keys can be connected to USB or parallel ports on the same server; up to five parallel port keys can be attached to the same parallel port. There is no limit to the maximum number of servers you can have on the network.

Thus, the network's total concurrent license limit is the sum of all the limits in all keys attached to all servers. If a user attempts to access a protected application (assuming the application is running in the default *dual mode*—see page 101), and the first server has reached its license limit, SentinelSuperPro automatically checks the first key on another server for an available license. Use of multiple servers helps avoid a single point of failure.

- **Application Time-Out**

The server can disconnect a user, and release the license for use by other users, after a pre-determined amount of time has elapsed without a SentinelSuperPro query or heartbeat message. This helps prevent idle users from tying up licenses, and permits recovery of licenses used by aborted programs or workstations that are unexpectedly disconnected from the network.

## What's New in SentinelSuperPro 6.1?

SentinelSuperPro 6.1 improves upon SentinelSuperPro 5.1 by adding new features and updating existing features. These features include:

- Support for up to 28 protected applications per key\*
- Network licensing capabilities allow for concurrent use of an application by multiple clients using a single key
- Integrated protection, key programming and activation functions in one application (the SSP Toolkit)\*
- Enhanced user interface\*
- Server monitoring tool allows system administrators to track license usage on the network
- Automatic cell allocation of elements\*
- Ability to shell multiple applications on the same key\*
- Addition of a one-time update option for license codes, allowing you to prevent a license code from being applied more than once during field exchange
- Activation actions/commands are automatically generated for applications that require activation\*
- Unused cells can be skipped, randomized or cleared during key programming\*
- Use of distributor keys provides developer control of the number of end-user keys distributors can activate and update
- Ability to override pre-programmed license limits through the use of sublicensing
- Integrated applications, shelled applications and custom elements can comfortably co-exist on the same key\*

---

**Note:** Features marked with an asterisk (\*) were originally available in SentinelSuperPro 6.0.

---



---

## What's Included with SentinelSuperPro 6.1?

The SentinelSuperPro 6.1 package includes:

1. The SentinelSuperPro 6.1 CD, with the following software:
  - SentinelSuperPro 6.1 Developer's Toolkit
  - SentinelSuperPro 6.1 Server (*loadserv.exe*, *spnsrv9x.exe*, *spnsrvNT.exe*)
  - SentinelSuperPro 6.1 Monitoring Tool (*monitor.exe*)
  - Sentinel system driver
  - Sentinel data protection driver
  - Make Keys Utility (*MakeKeysUtil.exe*)
  - License Generator Utility (*LicenseGenUtil.exe*)
  - Field Exchange Utility (*FieldExUtil.exe*)
  - Command-line shell utility (*ShellUtil.exe*)
  - Sentinel Client Activator
  - SentinelSuperPro merge modules for use with Windows Installer
  - SentinelSuperPro language interfaces
  - SentinelSuperPro documentation, including the *SentinelSuperPro 6.1 Developer's Guide* and the *SentinelSuperPro 6.1 System Administrator's Guide*.
  - Adobe Acrobat Reader (for accessing online documentation in PDF format)
2. One SentinelSuperPro hardware key

The type of key included in your package depends on whether you ordered a network or a non-network version of SentinelSuperPro. See page 27 for more information.

3. A document listing the passwords you need to program the key, including your developer ID, the write password and the overwrite passwords

---

---

**Warning!** *Your developer ID and passwords control access to your hardware key—do not lose them. If you do, you will need to return the key to Rainbow Technologies for a replacement. Also, to prevent unauthorized use of the key, be sure to keep the password document secure!*

---

---

4. The *SentinelSuperPro 6.1 Quick Start Guide*
5. Sentinel Client Activator documentation
6. A *readme.txt* file that provides late-breaking information about SentinelSuperPro, including system requirements, installation information and documentation updates (when necessary)

---

## System Requirements

Review the following hardware and software system requirements prior to installing the SentinelSuperPro Developer's Toolkit.

---

**Note:** *System requirements for using the SentinelSuperPro Server and Monitoring Tool can be found in the SentinelSuperPro 6.1 System Administrator's Guide.*

---

### Minimum Hardware Requirements

- Pentium microprocessor, P90
- VGA monitor (800 x 600 resolution recommended)
- 30 MB free hard disk space
- CD-ROM drive
- 32 MB RAM

### Minimum Software Requirements

- Microsoft® Windows® NT 4.0 Workstation with Service Pack 4 installed, Windows® 95, Windows® 98, Windows® ME or Windows® 2000
- Microsoft® Internet Explorer 4.01 or higher (to view the SentinelSuperPro online Help file)

Go to <http://www.microsoft.com> on the Web to install a free version of Internet Explorer.

---

**Note:** You **must** have Internet Explorer 4.01 or higher installed to be able to view the SentinelSuperPro online Help file. This file **cannot** be viewed with any other browser, such as Netscape Communicator.

---



# Chapter 2

---

## Installation

Before you can begin protecting your applications, you must install the SentinelSuperPro Developer's Toolkit, the SentinelSuperPro Server, the Sentinel system driver and the SentinelSuperPro hardware key.

Options for installing SentinelSuperPro client library interfaces, Visual C++ development tools and the SentinelSuperPro Monitoring Tool are also provided.

The following topics are covered in this chapter:

- Running SentinelSuperPro setup
- Installing the SentinelSuperPro hardware key
- Installing SentinelSuperPro interfaces
- Installing the Sentinel Client Activator

---

## Running SentinelSuperPro Setup

If you are installing SentinelSuperPro 6.1.1 on a Windows NT or 2000 workstation, you must have administrator-level access.

Before you begin installing SentinelSuperPro 6.1.1 components:

- **Save** and **Exit** out of all currently running applications.
- If you have SentinelSuperPro 6.1.0 installed, uninstall it. Versions 6.1.0 and 6.1.1 cannot co-exist on the same system.

---

**Note:** *If you have SentinelSuperPro 5.1 or 6.0 installed on your system, you do not need to uninstall it before you begin this procedure. SentinelSuperPro 6.1.1 can co-exist on the same system as 5.1 or 6.0 without any problems. See Chapter 16, “Migrating From SentinelSuperPro 5.1 or NetSentinel,” on page 321 for more information.*

---

- Disconnect all SentinelSuperPro USB hardware keys from your system.
- Verify that you have Internet Explorer (IE) 4.x or higher installed on your system.

Setup will not run if IE 4.x or higher is not installed; to install the latest version of IE, go to <http://www.microsoft.com>.

Internet Explorer is used to view SentinelSuperPro HTML Help. It must be installed prior to setup so that additional components required to view HTML Help files can be installed.

### Preparing to Install

1. Place the SentinelSuperPro 6.1.1 CD in your CD-ROM drive.

The setup program should start automatically. If not, navigate to the following path (assuming E: is the drive letter of your CD-ROM drive): *E:\Start.exe*, and then double-click on the file.

The SentinelSuperPro installation screen appears.

---

**Note:** *If you have problems accessing the SentinelSuperPro installation screen, you can start the setup program manually by navigating to the following path: E:\SentinelSuperPro6.1.exe, and then double-clicking on the file.*

---

2. Click **SentinelSuperPro 6.1 Developer's Toolkit**. A status bar appears while setup prepares to start the installation process.
3. Do one of the following:
  - If this is the first time you have run the Windows Installer program on this system, a message box prompting you to reboot your system may appear. If so, you must restart your system to continue the installation. Click **Restart**, then go to step 4.
  - If the above message box does not appear, go to step 5.
4. Once your system has completed rebooting, SentinelSuperPro setup should resume automatically. If it does not, double-click the setup file again to restart the installation process.
5. After the file is unpacked, do one of the following:
  - If the Sentinel Driver Upgrade screen appears, go to step 6.
  - If the SentinelSuperPro Welcome screen appears, go to step 1 of “Installing SentinelSuperPro Components” on page 22.
6. If you already have a previous version of the Sentinel driver installed, the Driver Upgrade screen appears. You *must* upgrade to the latest version of the Sentinel driver to use SentinelSuperPro 6.1. Read the on-screen message, then click **Upgrade**.

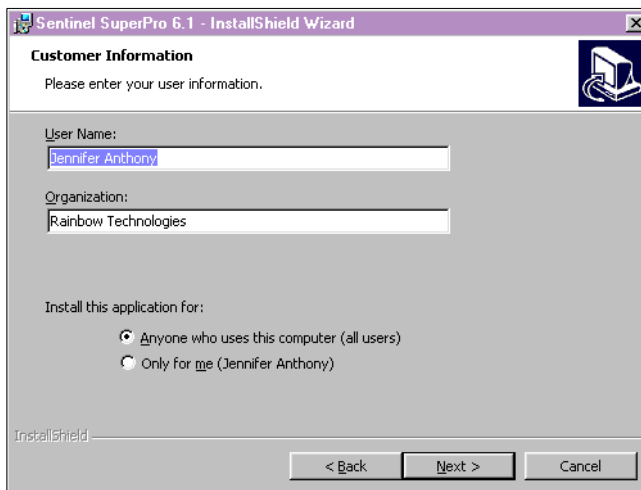
## Installing SentinelSuperPro Components

After the setup program has verified your system has the appropriate installer files and has checked for an existing Sentinel driver, you are ready to start installing SentinelSuperPro. The SentinelSuperPro Welcome screen appears.

1. After reading the preliminary information, click **Next**. The license information appears.
2. To accept the license and continue, select the **I accept the terms in the license agreement** option, then click **Next**.

The Next button is not available until you select the “I accept” option.

The following screen appears:



### Customer Information Screen

3. Enter or verify your user name and organization name in the appropriate fields.
4. Select who you want this application to be installed for.



- To allow anyone who logs on to this system to be able to run SentinelSuperPro, select **Anyone who uses this computer (All users)**.
- To make SentinelSuperPro accessible only when you are logged on to this system, select **Only for me (your name)**.

To increase the security of your application protection strategies, you may want to prevent other users from being able to run the SentinelSuperPro Toolkit by selecting **Only for me**. However, if more than one user will need access to SentinelSuperPro on this system, be sure to select **Anyone who uses this computer**.

5. Click **Next**. The Setup Type screen appears.

6. Select one of the following:

- **Complete**: Installs all SentinelSuperPro components, including the SentinelSuperPro 6.1 Developer's Toolkit, the SentinelSuperPro Server, the SentinelSuperPro Monitoring Tool, and other supporting files and utilities.

A complete installation requires 35 MB of free disk space.

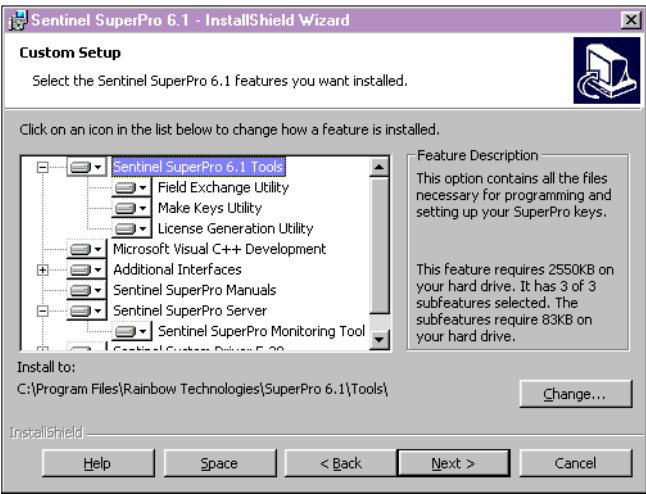
---

**Note:** *You cannot change the default installation location of C:\Program Files\Rainbow Technologies\SuperPro\6.1 when you select Complete. To change the installation location, select **Custom**.*

---


- **Custom**: Allows you to select which components you want to install, and to change the default installation location for each component.

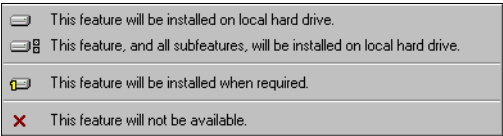
If you selected **Complete**, go to step 13. If you selected **Custom**, the list of SentinelSuperPro components appears. Go to the next step.



Custom Setup Screen – List of Components


**Note:** If this is your first time running the setup program, you must install **SentinelSuperPro 6.1 Toolkit**, **SentinelSuperPro Server** and **Sentinel System Driver 5.39**.

7. By default, all components are marked for installation. To select which components will not be installed, click the component’s icon . A shortcut menu appears.





Install Options Shortcut Menu

8. From the shortcut menu, select the appropriate option.

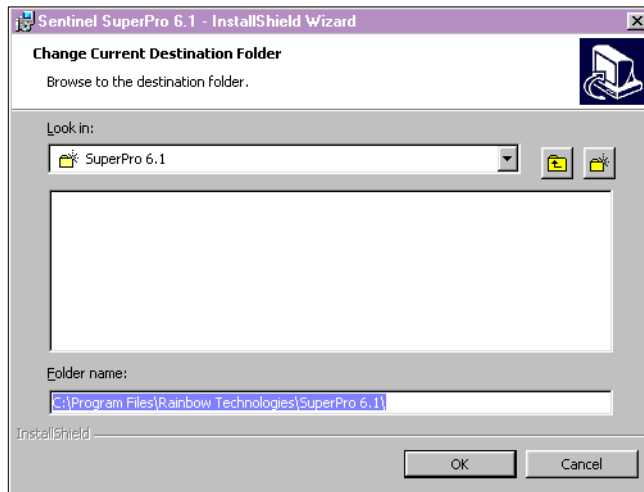
For example, to prevent a component from being installed, select **This feature will not be available**. An  appears in place of the component’s icon.

---

**Note:** If a  is located to the left of a component's icon, that component has subcomponents that can be selected for installation separately. Click the  to view the available subcomponents.

---

9. To change the location where SentinelSuperPro components will be installed, select a component, then click **Change**. The Change Current Destination Folder screen appears.



### Change Current Destination Folder Screen

10. To change the default location for the selected component, browse to select a new folder, then click **OK**.
11. Repeat steps 9 and 10 for each component you want to change the installation location for.

---

**Note:** ***You should always install the SentinelSuperPro Server on a local hard drive. Do not install the server on a network drive. If you install the server on a network drive, you will be unable to start it, and thus unable to use SentinelSuperPro.***

---

12. Once you have selected which components you want to install, click **Next**.
13. When the Ready to Install screen appears, click **Install**. SentinelSuperPro begins installing on your system. Once installation has finished, an “install complete” screen appears.

---

**Note:** *Depending on your operating system, you may need to reboot your system at this point. You will be prompted if a reboot is required; if a message appears, follow the on-screen instructions.*

---

14. Click **Finish**, then go to the next section to install the SentinelSuperPro hardware key.

---

## Installing the SentinelSuperPro Hardware Key

SentinelSuperPro comes with one hardware key for you to use while you are designing and implementing your application protection strategies. The key must be connected to your workstation while you run SentinelSuperPro software.

When you placed your order for SentinelSuperPro, you should have specified whether you wanted a network version, or a non-network version. The type of key you receive—network or stand-alone—depends on the version of SentinelSuperPro you ordered. Network keys can be identified by the phrase “SuperProNet” stamped into the plastic on one side of the key.

SentinelSuperPro hardware keys come in two form-factors: *parallel port* or *USB*. Again, the type you received in your package depends on what you specified when you placed your order.



**SentinelSuperPro USB Key (left) and Parallel Port Key (right)**

Parallel port keys (25-pin or 36-pin) connect to a parallel port located on the back of your computer. USB keys connect to a USB port located on the back or front of your computer or on a USB hub. Use the instructions in the appropriate following section to install your hardware key.

---

**Note:** For instructions on installing SentinelSuperPro hardware keys on a server, please refer to the SentinelSuperPro System Administrator's Guide.

---

## Installing the Parallel Port Hardware Key

The SentinelSuperPro key can be attached to any parallel port on your computer, as the Sentinel driver automatically polls each port to locate the key.

1. Locate an available parallel port on your computer.

If your computer has only one parallel port, you may need to temporarily remove any existing parallel port devices (such as a Zip drive or printer) in order to connect the key. These devices may be reconnected to the key's outside connector after you have installed the key.

2. Attach the key to the parallel port connector.
  - If you are using a 25-pin key, we recommend you attach the key directly to the parallel port without using an extension cable between the computer and the key. However, you may use a cable to connect a printer or other parallel device to the key; see below for more information on using cables with the SentinelSuperPro key.
  - If you are using a 36-pin key, you may use a cable to connect the key to the computer, but do not use an extension cable to connect a printer or other parallel device to the key.
3. Tighten the screws to connect the key securely to the port.
4. If necessary, reconnect any other parallel port devices to the outside connector on the key. We recommend using a shielded printer cable if you are connecting a printer to your computer through the SentinelSuperPro key.

### ***Using Cables with the SentinelSuperPro Hardware Key***

Due to the large variety of cables currently on the market, Rainbow Technologies does not recommend a specific brand or type of cable for use with the SentinelSuperPro key, nor do we guarantee that all cables will be compatible with the key.

However, we do recommend the following:

- Cables should not be longer than 6 feet in length.
- Cables should be shielded.
- Do not use ribbon cables.
- Cables must be straight-through; that is, they must have all pin signals wired through to the connectors on either end of the cable.

Please be aware that not all combinations of cables and printers are compatible with the SentinelSuperPro key—contact Rainbow Technical Support if you encounter a compatibility problem.

### **Connecting Multiple Parallel Port Keys**

Multiple SentinelSuperPro keys can be attached to the same parallel port; this is called *cascading*. Cascaded keys do not all need to have the same developer ID, and network keys can be cascaded with stand-alone keys.

The number of keys that can be cascaded is dependent on the system they are attached to. Typically, up to five keys can be connected to the same port. Refer to your system's documentation for more information about attaching devices to your parallel port.

SentinelSuperPro keys can also be cascaded with other Rainbow Technologies keys that support cascading. Make sure the SentinelSuperPro keys are the *last* keys in the chain (farthest from the computer).

---

---

**Warning!** *There is one exception to this rule. If you are cascading SentinelSuperPro keys with Rainbow Technologies' NetSentinel keys, the SentinelSuperPro keys **must be located before any NetSentinel keys** in the chain. NetSentinel keys should be the last keys in the chain. If SentinelSuperPro keys are located behind NetSentinel keys, they will not be recognized by the Sentinel driver, and thus the protected application will not run.*

---

---

While SentinelSuperPro keys can be cascaded with keys from other companies, this may cause compatibility issues and is not recommended.

When your application attempts to establish communication with a key, it must specify the developer ID. The driver then locates the first key with this developer ID. The application can also ask for another key with the same developer ID if the first key is not desired for some reason.

Keys can also be connected to up to three parallel ports on the same computer. For example, if you have three parallel ports on your computer, you could attach a separate key to each port simultaneously. The Sentinel driver automatically polls all parallel (and USB) ports when looking for a key.

---

**Note:** *You cannot cascade keys with the same developer ID while you are programming keys. Every key must be programmed individually.*

---

### Installing the USB Hardware Key

If you have multiple USB ports (if, for example, you are using a USB hub), you can connect up to 10 USB hardware keys on a single computer. Cascading—connecting multiple keys to the same port—is not supported for SentinelSuperPro USB hardware keys.

We recommend installing the Sentinel system driver prior to connecting any USB keys to your system.

1. Locate an available USB port on your computer.
2. Attach the key to the USB port. Make sure it is securely and tightly connected.

---

**Note:** *USB hardware keys can be used with Windows 98/ME or Windows 2000 workstations only.*

---



---

## Installing SentinelSuperPro Interfaces

Interfaces to the client libraries for several supported development languages are also accessible from the SentinelSuperPro installation screen. These interfaces provide examples for using the SentinelSuperPro API in the development language you are using to develop your applications.

---

**Note:** *The interfaces located on the CD are also available through the SentinelSuperPro installation program as a custom option.*

---

To install one or more language interfaces:

1. Place the SentinelSuperPro 6.1 CD in your CD-ROM drive.

The setup program should start automatically. If not, navigate to the following path (assuming E: is the drive letter of your CD-ROM drive):  
*E:\Start.exe*, and double-click on the file.

The SentinelSuperPro installation screen appears.

2. Click **Interface List**. A directory listing of the supported interfaces appears.
3. Double-click on the folder for the interface you want to install. A list of the interface files appears.
4. Copy and paste the files from the folder to your hard drive.
5. After you have saved all of the interface files to your hard drive, navigate to the directory you saved them in to access the *readme.txt* file. This file contains detailed information on using the specific interface you have downloaded.

---

## Installing the Sentinel Client Activator

Also available on the SentinelSuperPro installation screen is the Sentinel Client Activator installation file.

If you want to use the Client Activator with your SentinelSuperPro-protected applications, you should (but are not required to) install it before you begin designing your protection strategy. Use of the Client Activator is an optional feature.

To install the Client Activator from the SentinelSuperPro installation screen:

1. Place the SentinelSuperPro 6.1 CD in your CD-ROM drive.

The setup program should start automatically. If not, navigate to the following path (assuming E: is the drive letter of your CD-ROM drive):  
*E:\Start.exe* and double-click on the file.

The SentinelSuperPro installation screen appears.

---

**Note:** *If you have problems accessing the SentinelSuperPro installation screen, you can start the Client Activator setup program manually by navigating to the following path:  
**E:\CA21\setup.exe**, and then double-clicking on the file.*

---

2. Click **Client Activator**. The Client Activator installation program should start automatically.
3. Refer to the Client Activator documentation (included on the Sentinel-SuperPro installation CD) for further installation instructions.

# Chapter 3

---

## Using the Hardware Key

The hardware key is the heart of SentinelSuperPro protection. The key controls and verifies access to your protected applications, assuring that only authorized users can run them.

Before you begin designing your protection strategy, however, you should understand how the key works, and how it can be used.

This chapter covers the following topics:

- Physical key layout
- Possible uses for the key
- Reserved cells
- Access codes
- Cell values
- Cell types
- Algorithm values and addresses
- Ordering and returning keys

# Getting to Know the Key

Every SentinelSuperPro key contains 128 bytes of memory, organized as 64 *cells* (words) of 16 bits each. Cells are addressed as locations 0 through 3F hex.

Reserved Cells	00	01	02	03	04	05	06	07
	08	09	0A	0B	0C	0D	0E	0F
Available Cells	10	11	12	13	14	15	16	17
	18	19	1A	1B	1C	1D	1E	1F
	20	21	22	23	24	25	26	27
	28	29	2A	2B	2C	2D	2E	2F
	30	31	32	33	34	35	36	37
	38	39	3A	3B	3C	3D	3E	3F

SentinelSuperPro Key Memory Cell Layout

**Tip:** Think of a cell as being a holding container (memory location) for the words that make up your algorithms, counters and other elements. Cells have addresses that represent their location on the key, much like street addresses represent the location of houses in a neighborhood.

When you program a cell, you assign it various attributes. These attributes determine how the cell (and the word it contains) is used by your application. Cell attributes include the *cell type*, the *access code* and the *cell value*.

Each of these attributes are explained later in this chapter. Generally, each cell contains one of the following types of words:

- **Data Words:** A data word can store data such as sublicenses, customer information, serial numbers, passwords, and check digits. You code your application to read the word and then evaluate and act upon the stored value. A data word cell may be programmed as read-only or read/write.
- **Counter Words:** A counter word contains an initial value you set that is then decremented by your application. A typical use of a

counter word is to limit the number of times a demo application can be executed.

- **Algorithms:** An algorithm contains a bit pattern that defines how the hardware key should encrypt query data sent by your application. The key uses an algorithm—plus an internally stored proprietary algorithm—to transform the query data and then return a value to your application. You design your application to send queries to the key and then evaluate and act upon the responses.

Algorithms are *active* or *inactive*. Only active algorithms can be return a valid response to a query. The *active/inactive bit* in the cell value controls whether or not the algorithm is active. See “Algorithm Values” on page 45.

Additionally, all algorithms are two words (and thus, two cells) long, and may have activation passwords and counters associated with them (see “Valid Algorithm Addresses” on page 47).

## Restricted Cells

Cells 00 through 07 in each key are restricted cells that contain fixed, pre-programmed system information:

**SentinelSuperPro Key Restricted Cells Contents**

Cell	Contents	Readable?
00	Key serial number; sequentially assigned per key. <sup>a</sup>	Yes
01	Developer ID; unique to your company/product.	Yes
02 – 07	Reserved for use by Rainbow Technologies.	No

- Maximum 16-bit value (0 – 65535). Serial numbers are not guaranteed to be unique. If you require unique serial numbers, please contact your Rainbow sales representative, as Rainbow must program the keys.

## Programmable Cells

Cells 08 through 3F are available for you to program. The next section explains in detail how to program these cells in the key.

---

## Programming the Key

When you program the key, you are actually assigning attributes to the cells. These attributes describe how the words contained in the cells are used to protect your application. There are three cell attributes: *cell type*, *access code* and *cell value*. Each of these attributes are explained in the following sections.

### Access Codes

Every cell has an *access code* associated with it that controls how the cell can be used by your application—it defines the cell's cell type attribute. For example, some cell types have an access code that permits cell values to be both read and overwritten, while others are read-only, or not writable at all. Access codes are numbers from 0 to 3.

When you define an element using the SSP Toolkit's Element Definition Wizard, you do not assign the cell access codes. The access codes are determined by the wizard, based on the protection feature you are implementing. If your application programs or reprograms cells in the field, it must specify the new access code.

The following table describes the four available access codes:

SentinelSuperPro Key Cell Access Codes

Code	Description
0	<b>Read/write data word</b> Your application can read the word in the cell and, if the write password is supplied, modify its contents.
1	<b>Read-only (locked) data word</b> Your application can read the word in the cell, but cannot change it without the overwrite passwords.
2	<b>Counter word</b> The cell contains a word (value) that your application can decrement using the write password. The cell's value cannot be changed (other than by decrementing it) without the overwrite passwords.
3	<b>Locked and hidden/algorithm word</b> Your application cannot read the cell's value. Modification requires the overwrite passwords. The cell value (contents) is hidden (unreadable).

Cell Types

Each cell is assigned a code that defines how you want to use the selected cell. This code is called a *cell type*. The cell type classifies the type of data stored in the cell, which in turn affects how the cell can be used.

Each cell type is identified by a two-letter abbreviation; for example, CW identifies a counter word.

Some cell types are designed to be used in groups. For example, algorithms can have counters and passwords associated with them. Other cell types have *address restrictions*, meaning they can be assigned only to specific cells on the key.

The following table describes the available cell types, with the following sections explaining each cell type in greater detail.

SentinelSuperPro Key Cell Types

Cell Type	Access Code	Name
**	0	Undefined
AA	3	Active Algorithm
AH	3	Algorithm Half
AP	3	Activation Password
CA	2	Algorithm Counter Word
CW	2	Counter Word
DI	1	Developer ID
DL	1	Locked Data Word
DW	0	Data Word
IA	3	Inactive Algorithm
RW	3	Reserved Word
SN	1	Serial Number

**Undefined (\*\*)**

The Undefined cell type is used to identify a cell that has not yet been programmed or is not used in your protection strategy. This cell type is identified by two asterisks (\*\*).

Cells you don’t need for your protection strategy can be left undefined. However, you may want to program unused cells as read-only data words or algorithm/hidden words. This prevents them from being accessed.

---

**Tip:** Undefined cells can also be programmed with random values to make your strategy more confusing for hackers. See “Programming a Product Key” on page 217 for more information.

---



## Access Code

An Undefined cell has an access code of 0 – read/write data.

## Valid Addresses

Any unrestricted cell (08 – 3F) can be classified as Undefined.

## ***Active Algorithm (AA)***

The Active Algorithm (AA) cell type defines an active (enabled) algorithm. An algorithm consists of two adjacent AA cells (words) with access codes of 3. The values in these cells affect the way query data is encrypted via the RNBOsproQuery() API function. An algorithm must be active for it to return a valid response to a query.

The value in the second AA cell must be between 8000 and FFFF. See “Algorithm Values” on page 45 for more information.

AA cells can have a password and counter(s) associated with them.

## Access Code

An AA cell has an access code of 3 – algorithm/hidden.

## Valid Addresses

The first AA word must be in a cell located at an unrestricted, even address. Additional restrictions apply if a counter and/or password is associated with the algorithm. See “Valid Algorithm Addresses” on page 47 for more information.

## ***Algorithm Half (AH)***

The Algorithm Half (AH) cell type can be used for each of the two cells required for an algorithm. The algorithm created by two AH cells is basically the same as that created by two AA or IA cells; the difference is that you can program the descriptor in two steps, which may be useful in some protection schemes.

The value in the second AH cell must be between 0000 and 7FFFF (for an inactive algorithm) or 8000 and FFFF (for an enhanced algorithm). See “Algorithm Values” on page 45 for more information.

AH cells can have a password and counter(s) associated with them.

---

**Note:** *Use of the AH cell type requires a thorough understanding of algorithms. Consider using the AA and IA cell types instead.*

---

### Access Code

An AH cell has an access code of 3 – algorithm/hidden.

### Valid Addresses

An AH word can be located in any unrestricted cell (08 – 3F). You must leave an adjacent cell vacant for the other half of the algorithm. Also, the first AH word of the pair must be located in an even-numbered cell.

Additional restrictions apply if a counter and/or password is associated with the algorithm. See “Valid Algorithm Addresses” on page 47 for more information.

### Activation Password (AP)

The Activation Password (AP) cell type is used to activate an inactive algorithm so it can be used for queries. This allows activation, via a password, of an algorithm at a customer’s site. For detailed instructions on using activation passwords, see “Using Activation Passwords” on page 76.

The AP cell type must be two cells long and must immediately follow the algorithm it activates.

### Access Code

An AP cell has an access code of 3 – algorithm/hidden. It cannot be directly read or written to; its value is used only to verify a user-supplied password during execution of the `RNBOsproActiveAlgorithm()` API function.

Because an AP cell has an access code of 3, it can also be used as an algorithm. See “Querying Activation Passwords” on page 87 for more information.

## Valid Addresses

An AP cell must be located immediately after a two-word algorithm (cell type AA, AH or IA).

Additional restrictions apply if a counter is also associated with the algorithm. See “Valid Algorithm Addresses” on page 47 for more information.

## ***Algorithm Counter Word (CA)***

The Algorithm Counter Word (CA) cell type defines a counter that deactivates an associated algorithm when the counter reaches zero. You program an initial value into the counter, then decrement it using the `RNBOsproDecrement()` API function. The CA cell must immediately precede the algorithm it deactivates.

Thus, this cell type can be used to control the number of times an application can be executed. See “Controlling Demo Applications” on page 90 for more information.

Optionally, you can associate two counters (two CA cells) with one algorithm. In this case, the first counter to reach zero deactivates the algorithm. If desired, you could use the second counter after the algorithm is re-activated with a password.

## Access Code

A CA cell has an access code of 2 – counter. It can be read, but cannot be written to except by the `RNBOsproDecrement()` or `RNBOsproOverwrite()` API functions.

## Valid Addresses

A CA cell is always located immediately before a two-word algorithm (cell type AA, AH or IA). See “Valid Algorithm Addresses” on page 47 for more information.

### **Counter Word (CW)**

The Counter Word (CW) cell type is used for a counter that is *not* used to deactivate an algorithm. You program an initial value into the counter word, then decrement it using the `RNBOsproDecrement()` API function. You code your application to check the value in the counter and proceed accordingly if the value reaches zero.

A CW cell is similar to a data word—it is used for storing data or keeping track of something such as a number of uses. It is more restricted than a data word, however, because you cannot overwrite it without the overwrite passwords. You can decrement it with only the write password.

One use of this cell type is to control specific functions within your application. For example, if you associate a counter with the Save button control, you can code the application so that when the counter reaches zero, the Save button will no longer be available, preventing the user from saving their work and making the application unusable in a practical sense.

### **Access Code**

A CW cell has an access code of 2 – counter. It can be read, but cannot be written to except by the `RNBOsproDecrement()` or `RNBOsproOverwrite()` API functions.

### **Valid Addresses**

Any unrestricted cell (08 – 3F) can be classified as a CW cell type.

---

---

**Warning!** *If you program a counter cell, and you use the next two cells for an algorithm, the counter will function as an algorithm counter. When the counter reaches zero, the algorithm will be deactivated, even if you did not intend for that to happen.*

---

---

### **Developer ID (DI)**

The Developer ID (DI) cell type is used for cell 01 only. This cell holds a read-only data word that contains the unique developer ID assigned to you by Rainbow Technologies. You cannot assign cell type DI to any other cell.

## Access Code

A DI cell has an access code of 1 – locked. You can read the developer ID, but cannot change it.

## Valid Addresses

The only cell that can be defined as cell type DI is cell 01.

## ***Locked Data Word (DL)***

The Locked Data Word (DL) cell type is used for data words you want your application to read, but not write to, such as sublicense cells.

## Access Code

A DL cell has an access code of 1 – locked. After you program the cell, your application can read it, but cannot change it without the overwrite passwords.

## Valid Addresses

Any unrestricted cell (08 – 3F) can be classified as a DL cell type.

## ***Data Word (DW)***

The Data Word (DW) cell type can store any value (data word) you want to use in your protection strategy. This value can be read and/or changed by your application. It can also be decremented.

## Access Code

A DW cell has an access code of 0 – read/write. It can be reprogrammed using the write password.

## Valid Addresses

Any unrestricted cell (08 – 3F) can be classified as a DW cell type.

### ***Inactive Algorithm (IA)***

The Inactive Algorithm (IA) cell type defines an inactive (disabled) algorithm.

An algorithm consists of two adjacent cells with access codes of 3. The values in these cells affect the way an input string is encrypted via the RNBOsproQuery() API function. An inactive algorithm cannot be used for a query until it is activated by the RNBOsproActivate() API function.

The value in the second IA cell must be between 0000 and 7FFF. See “Algorithm Values” on page 45 for more information.

IA cells should always have a password associated with them so the algorithm can be activated. They can also have one or two counters.

#### **Access Code**

An IA cell has an access code of 3 – algorithm/hidden.

#### **Valid Addresses**

The first IA cell must be at an unrestricted, even address. Additional restrictions apply if a counter and/or password is associated with the algorithm. See “Valid Algorithm Addresses” on page 47 for more information.

### ***Reserved Word (RW)***

The Reserved Word (RW) cell type is used for cells 05 through 07 only. These cells hold hidden words that are reserved for use by Rainbow Technologies. You cannot assign cell type RW to any other cell.

#### **Access Code**

An RW cell has an access code of 3 – algorithm/hidden. You cannot read or write to these cells.

#### **Valid Addresses**

The only cells defined as type RW are cells 05, 06 and 07.

## **Serial Number (SN)**

The Serial Number (SN) cell type is used for cell 00 only. This cell holds a read-only data word that contains the hardware key's serial number. The value in this cell is preprogrammed and cannot be modified. You cannot assign cell type SN to any other cell, or overwrite the SN cell.

---

**Note:** *Serial numbers range from 0–65535. They are assigned sequentially and are not guaranteed to be unique. If you require unique serial numbers, please contact your Rainbow Technologies sales representative, as Rainbow must program the keys.*

---

## **Access Code**

An SN cell has an access code of 1 – locked. You can read the serial number but cannot modify the value in this cell.

## **Valid Addresses**

The only cell defined as type SN is cell 00.

## **Cell Values**

Each cell also has a *cell value* containing a 16-bit value. The cell value is also known as a *word*. The value in the second cell of an algorithm controls whether or not the algorithm is active, and whether the enhanced algorithm engine is enabled for the algorithm. See the next section for more information.

## **Algorithm Values**

There are special rules applied to the second cell of an algorithm. The word (value) in the second cell controls:

- Whether the algorithm is active or inactive. Only active algorithms can be used for queries.
- Whether the enhanced algorithm engine is enabled or disabled. The enhanced algorithm engine provides a more secure algorithm.

The active/inactive state of an algorithm is controlled by bit 7 of the second word of the algorithm:

- If this bit is 1, the algorithm is active.
- If this bit is 0, the algorithm is inactive

The state of the enhanced algorithm engine is controlled by bit 6 of the second word of the algorithm:

- If this bit is 1, the enhanced engine is enabled.
- If this bit is 0, the enhanced engine is disabled.

The following tables show how bits 6 and 7 of the second word control the algorithm.

**Second Word of an Active Algorithm, with Enhanced Engine Enabled**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	0	0	0	0	0	0

**Second Word of an Active Algorithm, with Enhanced Engine Disabled**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	0	0	0	0	0

**Second Word of an Inactive Algorithm, with Enhanced Engine Enabled**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	0	0	0	0	0

**Second Word of an Inactive Algorithm, with Enhanced Engine Disabled**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0



When you design your protection strategy, SentinelSuperPro asks you if you want the algorithm to be active or inactive. The following table summarizes the effect of the value of the second word on the algorithm:

Possible Values for an Algorithm’s Second Word

	Enhanced Engine Disabled	Enhanced Engine Enabled
Algorithm Inactive	0000 – 3FFF	4000 – 7FFF
Algorithm Active	8000 – BFFF	C000 – FFFF

For example, an algorithm with a second word of **1FDC** is inactive and has the enhanced engine disabled because it falls within the range of 0000 – 3FFFF. An algorithm with a second word of **D000** is active and has the enhanced engine enabled because it falls within the range of C000 – FFFF.

---

**Note:** *For maximum security, use of the enhanced algorithm engine is recommended for all algorithms.*

---

### Valid Algorithm Addresses

Certain cell types are designed to be used only in groups. These cell types—Activation Passwords (AP) and Algorithm Counters (CA)—are used only in association with algorithms (cell types AA, AH and IA).

---

**Tip:** *While you are designing your protection strategy, the Element Definition Wizard only allows you to select valid, available addresses for your elements. You don’t need to worry about these restrictions while you are adding elements, but you should still be aware they exist.*

---

These groups of cells are restricted as to where they can be placed on the hardware key. The following combinations of algorithms, counters and passwords (known as *custom elements*) are supported:

- Algorithm (2 cells)
- Algorithm with password (4 cells)

- Algorithm with counter (3 cells)
- Algorithm with two counters (4 cells)
- Algorithm with password and counter (5 cells)
- Algorithm with password and two counters (6 cells)

---

**Tip:** Remember, each word takes up one cell. So an algorithm with two counters uses four cells, because it has four words: two counters, plus the algorithm’s two words.

---

SentinelSuperPro automatically selects appropriate locations for your algorithms when you add a custom element in the Design stage. You also have the option to select the locations yourself in Element Layout View, but you are not allowed to place the algorithm in an invalid position. See Chapter 8, “Working With Design Elements,” on page 169 for more information about adding custom elements to your protection strategy.

The address restrictions for these cell groups are summarized in the following sections. In this discussion, an algorithm (identified by ALGO in the following tables) can be defined using AA, IA or AH cells.

---

**Note:** MOD is used in the formula used to compute valid cell addresses. MOD is a modulus arithmetic operator used to divide two numbers, resulting in the remainder of the division. For example, 8 MOD 3 equals 2 because 8 / 3 equals 2, with a remainder of 2. The remainder is the result of the MOD notation.

---

### How to Read the Tables in the Following Sections

In the next sections, tables are used to show you the valid locations for each type of element. Use the following legend while reading these tables:

Color	Element
Blue	Algorithm
Magenta	Counter
Yellow	Password

## Algorithm

A two-word algorithm that does not have a counter or password. It can start in any unrestricted cell with an even address.

**Valid Algorithm Locations**

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

## Algorithm with Password

A two-word algorithm that has an activation password (AP). It must start in a cell with an address equal to 0 MOD 4. The two-word password must immediately follow.

**Valid Algorithm with Password Locations**

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

**Algorithm with Counter**

A two-word algorithm that has one counter (CA). The algorithm must start in a cell with an address equal to 4 MOD 8. The counter word must immediately precede the algorithm.

**Valid Algorithm with Counter Locations**

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

The relationship between a counter word and an adjacent algorithm exists even if you do not intentionally plan it. The algorithm will be deactivated when the counter reaches zero.

**Algorithm with Two Counters**

A two-word algorithm that has two counters (CA). The algorithm must start in a cell with an address equal to 4 MOD 8. The counter words must immediately precede the algorithm.

**Valid Algorithm with Two Counter Locations**

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

### ***Algorithm with Password and Counter***

A two-word algorithm that has both a counter (CA) and an activation password (AP). The algorithm must start in a cell with an address equal to 4 MOD 8. The counter word must immediately precede the algorithm, and the two-word password must immediately follow it.

**Valid Algorithm with Password and Counter Locations**

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

### ***Algorithm with Password and Two Counters***

A two-word algorithm with two counters (CA) and an activation password (AP). The algorithm must start in a cell with an address equal to 4 MOD 8. The counters must immediately precede the algorithm and the two-word password must immediately follow it.

**Valid Algorithm with Password and Two Counters Locations**

00	01	02	03	04	05	06	07
08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27
28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37
38	39	3A	3B	3C	3D	3E	3F

---

## Ordering and Returning Keys

The SSP Toolkit comes with one SentinelSuperPro hardware key (see page 27). You need to order additional keys that you will program and then ship with your protected applications.

### Ordering Additional Keys

Contact your Rainbow Technologies representative to order additional SentinelSuperPro hardware keys. When you place your order, be sure to have your developer product identifier available. This code can be found on the hardware key itself. Each key's label shows:

- **A developer product identifier:** This code is assigned to all keys you use to protect a specific product. The first character is always “S,” identifying the key as a SentinelSuperPro key. The next two characters identify the distributor and the last six characters are the sequence numbers.

---

**Note:** *The developer product identifier is **NOT** the same as the developer ID programmed into the key. The product identifier is used for ordering purposes only.*

---

- **A manufacturing code:** This code identifies the key's manufacturing lot. It is helpful to Rainbow Technologies if a key is returned for any reason.



**SRB006439** — Developer  
**9938L24319** — Product Identifier  
— Manufacturing Code

**Sample SentinelSuperPro Key Label**

---

**Note:** *The developer product identifier and the manufacturing code are in the same format on both network and stand-alone keys.*

---

When you place your order, you should also have the following information on hand:

- What type of keys you want: *network* or *stand-alone*
- What form-factor you want your keys to be in: *parallel* or *USB*
- If you are using network keys, what you want for the hard limit (see page 3) that will be pre-programmed into the key

For more information about hardware key versions, see “Network Keys v. Stand-alone Keys” on page 3. For more information about key form-factors, refer to “Installing the SentinelSuperPro Hardware Key” on page 27.

### Returning Keys

Occasionally, you may find that you need to return a Rainbow product for exchange or repair. To ensure proper handling is acknowledged for the returned keys, you must obtain a Return Material Authorization (RMA) number prior to shipping the products to Rainbow. To obtain an RMA number:

- If you suspect a technical problem, call Rainbow’s Technical Support. The support representative will work with you to rule out resolvable software and/or configuration problems. If the problem cannot be resolved, the RMA department will assign you an RMA number over the phone.
- If you have keys to be returned for other than a specific technical situation, call Rainbow’s RMA department for an RMA number. Contact information can be found on page ii.

### ***Packaging the Keys for a Return***

After you have obtained an RMA number and are ready to package the keys for shipping, please read and follow these packaging guidelines:

- Install an electrostatic-dissipating mat as a work surface, and make sure the mat is properly grounded.
- Wear grounding wrist or ankle straps while handling the keys.
- Use packaging materials designed to avoid electrostatic charge during shipment. Plastic that does not generate static (“cold plastic”) is typically pink in color. You may also use “conductive plastic”, which is designed to drain off static.

---

---

**Warning!** *Electrostatic charges can damage the SentinelSuperPro keys. We strongly recommend following these guidelines at all times to prevent damage to your keys.*

---

---

Be sure to write your RMA number on the shipping label to ensure prompt and correct handling.



# Chapter 4

---

## Designing Your Protection Strategy

The goal of any software protection strategy developed using SentinelSuperPro is to significantly reduce the chance that someone can defeat the protection and use your application without the hardware key. In general, the time and expense required for a skilled hacker to break your scheme is directly related to the number and complexity of the locks you place in your application. Protection can be as simple or as complex as you wish.

Before you actually start adding protection to your source code, however, you should design your protection strategy, deciding what type(s) of protection you'll use, which activation types you need, and more.

This chapter introduces you to the types of protection you can use with SentinelSuperPro, gives some guidelines for using various protection types, and describes advanced protection techniques you can use for even greater security. Once you have read through this chapter, you will be ready to start the SSP Toolkit and begin adding protection to your application.

This chapter covers the following topics:

- Introduction to software security
- Protection types
- Activation types

- Using network licensing
- Guidelines for using various protection and activation types
- Controlling demo applications
- Reading stored data
- Using algorithms for encryption
- Advanced protection techniques
- Programming the key

---

**Note:** *All values and cell addresses used in this chapter's examples are in hexadecimal format. Also, for simplicity, standard error-checking steps are omitted from the examples. If you receive an invalid response to a query or another function, **we recommend retrying the operation before taking action.***

---

---

# Introduction to Software Security Concepts

Before you start making decisions about the strategy you're going to use to protect your applications, it's important that you understand the concepts behind software security.

## Protection Types

SentinelSuperPro provides two general types of protection: *integrated* and *automatic*. The protection type determines when and where software locks are implemented.

### ***Integrated***

When you choose integrated protection, you add software locks—API functions to verify the presence of the key—directly into your application's source code. You control the amount and location of the locks.

The frequency of software locks within your application, and the action taken if no key is found, is left up to you. The more locks you add to your application, the more difficult it will be for potential hackers to break your application's protection.

Because you must understand the API calls used to support the protection strategy you have designed, and manually add them to your code, using integrated protection may take longer.

Integrated protection is most commonly used when:

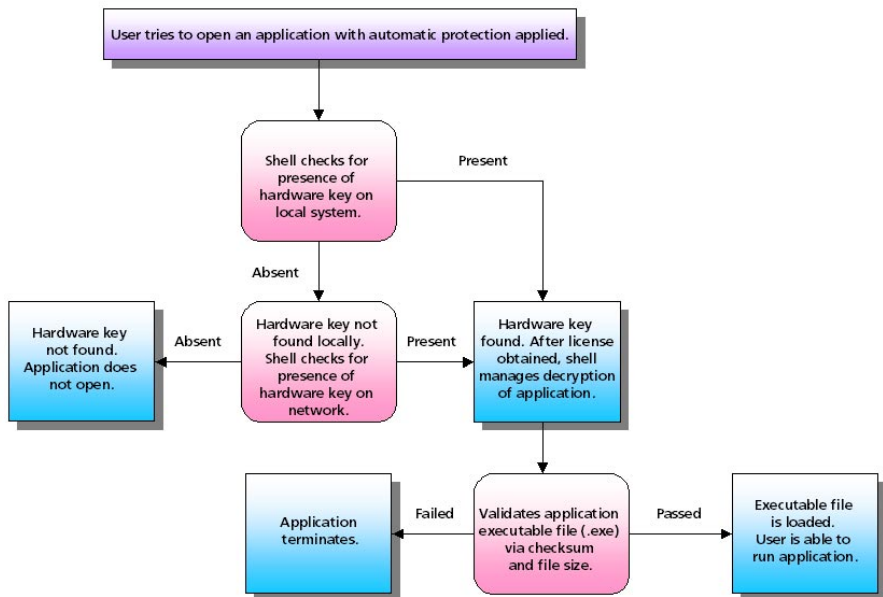
- You want to have control over the protection techniques used to secure your application.
- You have access to the source code and understand the API functions (see Chapter 15 for more information about the SentinelSuperPro API).

## Automatic (Shelled)

Automatic protection is a simplified, fast, and easy way to protect your application against unauthorized access.

When you use automatic protection, SentinelSuperPro wraps a protective layer, called a *shell*, around your application's executable file. This layer is encrypted, making it more difficult for a hacker to gain access to your application's code.

The shell layer makes no changes to your application's source code, so there is no need to recompile. All software locks and communication with the hardware key (such as checking and verification) is handled by the shell. An application protected with a shell can be run only if the user has the correct hardware key.



**How an Automatic Shell Protects an Application**

Automatic protection is most desirable when:

- You don't have access to the application's source code.
- You don't have the time or desire to design a unique protection scheme.
- You want additional security by implementing a shell in conjunction with API functions incorporated in your code.
- You want to effortlessly include time/date/execution controls for a demo application, without having to manually add the elements and API functions in code.

---

**Note:** *Some applications or .DLLs (such as those using threaded local storage) may not work correctly with automatic protection applied. In this case, use integrated protection instead.*

---

### Active v. Inactive Applications

As part of your protection strategy, you choose to make your application either *active* or *inactive*.

An **active application** is one that is ready to run when shipped to your customer. It will always remain active, as long as the hardware key is attached.

An **inactive application** will not run until it is activated.

For example, you can make your main application active, but make additional features inactive. This allows your customer to easily purchase and immediately activate upgrades in the field, because you can provide the activation password for the additional features without shipping additional software or visiting the customer's site.

Demo or metered applications are a special case, in that they are shipped as active, but usually become inactive after a specific number of executions. See "Controlling Demo Applications" on page 90 for more information.

## Activation Types

When you protect your applications with SentinelSuperPro, you also choose *how* you want your customer to activate future applications, or additional features, after installation.

The methods defining how customers activate your application are called *activation types*. There are four activation types in SentinelSuperPro: *active*, *static*, *trusted* and *distributed*.

The following table describes each of the available activation types, what you must do to use each type, and suggestions for how you can use each type. Typically, the activation type you use is based on whether you want your application to be active or inactive.

**SentinelSuperPro Activation Types**

Activation Type	Description	When to Use
<b>Active</b>	<ul style="list-style-type: none"> <li>Your application is <b>always</b> active when the hardware key is attached. It needs no activation password.</li> </ul>	<ul style="list-style-type: none"> <li>You want your main product to be always active so your customer can always run it.</li> <li>You might ship add-on features (that you intend to charge separately for) as inactive products, to be activated at a later time when your customer purchases them.</li> </ul>
<b>Static</b>	<ul style="list-style-type: none"> <li>The application is <i>inactive</i> until activated with an activation password, unless it is a demo or metered application.</li> <li>The activation password is the same for every hardware key used to protect the application. This means one password works for multiple keys.</li> </ul>	<ul style="list-style-type: none"> <li>This type is easier to deploy, because the password is always the same, making it easier to update several keys on different computers.</li> <li>If you are writing a separate activation password utility, you must use this type because you know what the password will be.</li> </ul>

## SentinelSuperPro Activation Types (Continued)

Activation Type	Description	When to Use
<b>Trusted</b>	<ul style="list-style-type: none"> <li>The application is <i>inactive</i> until activated with an activation password, unless it is a demo or metered application.</li> <li>Activation passwords are generated by SentinelSuperPro and are unique for each hardware key and each application.</li> <li>Requires distribution of the Field Exchange Utility or the Sentinel Client Activator for field activation.</li> </ul>	<ul style="list-style-type: none"> <li>Provides excellent security, because all passwords are unique.</li> <li>Best for use with applications using automatic (shelled) protection.</li> <li>You cannot use this type when you are writing your own activation password utility, because you never know what the password for a specific key will be.</li> </ul>
<b>Distributed</b>	<ul style="list-style-type: none"> <li>The application is <i>inactive</i> until activated by a product distributor, unless it is a demo or metered application.</li> <li>Activation passwords are generated by SentinelSuperPro and are unique for each hardware key and each application.</li> <li>Distributor uses the Field Exchange Utility or Sentinel Client Activator to activate the application. Each activation decrements the distributor key's counter.</li> <li>Requires programming and distribution of a distributor key in addition to the product keys.</li> </ul>	<ul style="list-style-type: none"> <li>Must be used if you want to keep track of the number of product activations performed by your distributors.</li> <li>If you want to charge your distributors for product activations. The distributor key keeps track of the number of activations, and when the counter reaches zero, no more activations are allowed. You can update (and charge for) a distributor key with more activations in the same way that product keys are updated.</li> </ul>

### ***Example: Using the Trusted Activation Type***

The following simple example is designed to give you a feel for how the trusted activation type is used. The trusted activation type is the most often used, and is appropriate for most protection strategies.

Assume you want to create 100 copies of your application to ship to your customers. Your product line is defined as follows:

- You have a main application—named *SceneryEditor*— that you want to run immediately at the customer's site.
- You have a demo of *SceneryEditor* that you want to run immediately at the customer site, but it will expire at some point.
- You have ten other add-on features for *SceneryEditor* that your customer may purchase in the future as upgrades.

In general, the following procedure describes how you would use SentinelSuperPro to protect and distribute *SceneryEditor* and its add-on features:

1. Use the SSP Toolkit to define the following:
  - Assign *SceneryEditor* an **active** activation type, by applying either integrated or automatic application protection.
  - Assign the demo version a **trusted** activation type.
  - Assign each of the add-on features (separately) a **trusted** activation type. Each add-on feature is treated as a separate application by SentinelSuperPro.
2. Add the protection to your applications by using the pseudocode protection plan generated during the **Implementation** stage to add the appropriate calls to your source code, or add the shell(s) to the appropriate executable files.
3. Produce 100 copies of *SceneryEditor* and its user documentation.
4. Program 100 SentinelSuperPro hardware keys with the protection strategy you defined in step 1, using the **Make Keys** stage.



5. Ship SceneryEditor, along with one product key and the Sentinel Client Activator or Field Exchange Utility.

If your customer decides to upgrade from the demo version, or decides to purchase additional features for SceneryEditor, he uses the Client Activator or Field Exchange Utility to obtain a *locking code*. Your customer sends you the locking code, and you do the following:

1. Start the SSP Toolkit and navigate to the **Implementation** stage.
2. Click the **Field Activation** tab, and then click **License Generator**.

---

**Note:** *You can also use the License Generator Utility to generate a license code.*

---

3. Enter the locking code provided by your user, and select the actions you want to perform on the key in the field.

The actions you select determine which features the user will have access to. SentinelSuperPro generates a *license code* that will apply the selected actions to the user's key.

4. Send the license code to your user, who then enters the code in the Client Activator or Field Exchange Utility, automatically activating the appropriate product or upgrade.

### ***Example: Using the Distributed Activation Type***

The next example is designed to give you a feel for how the distributed activation type is used. The distributed activation type is used when you want to give your sales distributors the ability to activate or update your application.

Assume you want to create 50 copies of your application to ship to customers, but these copies will be shipped to your distributors to sell. So that you can keep track of how many products your distributors sell, you decide to use the distributed activation type. Your product line is defined as follows:

- You have a main application—named *SceneryEditor*—that the distributor must activate in the product key before shipping it to the customer.
- You have three other add-on features for *SceneryEditor* that customers may purchase in the future as upgrades.
- You have five distributors, and you want each distributor to sell a maximum of 10 copies of *SceneryEditor*.

In general, the following procedure describes how you would use Sentinel-SuperPro to protect and distribute *SceneryEditor* through your distributors.

1. Use the SSP Toolkit to define the following:
  - Assign *SceneryEditor* a **distributed** activation type, by applying either integrated or automatic application protection.
  - Assign each of the add-on features (separately) a **distributed** activation type. Each add-on feature is treated as a separate application by SentinelSuperPro.
2. Add the protection to your applications by using the pseudocode protection plan generated during the **Implementation** stage to add the appropriate calls to your source code, or add the shell(s) to the appropriate executable files.
3. Produce 50 copies of *SceneryEditor* and its user documentation.

4. Program 50 SentinelSuperPro hardware keys with the protection strategy you defined in step 1, using the **Make Keys** stage.
5. Program a distributor key (with the protection strategy you defined in step 1) for each of your distributors who will be selling SceneryEditor, using the **Make Keys** stage.

The activation counter for each key should be set to 40—10 licenses to activate SceneryEditor, and 10 licenses each to activate each of the add-on features.

6. From the **Project** stage, export the protection strategy you defined in step 1 to a .DST file.
7. Ship the following items to each distributor:
  - Ten copies of SceneryEditor
  - Ten programmed product keys that have not yet been activated
  - Ten copies of the Sentinel Client Activator or Field Exchange Utility for distribution to customers with SceneryEditor
  - One of the distributor keys you programmed in step 5
  - The .DST file you created in step 6
  - The License Generator Utility

After the customer receives the application from the distributor, or if he decides to purchase additional features for SceneryEditor, he must activate the application or features before he can use them. To do so, he uses the Client Activator or Field Exchange Utility to obtain a *locking code*. The customer sends the distributor the locking code, and the distributor then does the following:

1. Connects his distributor key to his workstation.
2. Verifies that the SentinelSuperPro server is installed and running on his workstation.

3. Starts the **License Generator Utility** and opens the .DST file you provided with the protected application.
4. Enters the locking code provided by the customer, and selects the actions to perform on the key in the field.

The actions the distributor selects determine which features the customer will have access to.

5. Generates a *license code* that will apply the selected actions to the customer's key.

Each time a license code is generated, the activation counter in the distributor's key is decremented by one. Once the counter reaches zero, the distributor can no longer activate or update SceneryEditor.

---

**Note:** *The activation counter is decremented only when a distributor generates a license code to activate or update a distributed application. Distributors can activate or update an unlimited number of static or trusted applications; the activation counter is not decremented by these types of applications. In fact, distributor keys do not need to be connected at all in order to generate license codes for static or trusted applications.*

---

6. Sends the license code to the customer, who then enters the code in the Client Activator or Field Exchange Utility, automatically activating the appropriate product or upgrade.

When the license limit counter on a distributor's key reaches zero, you can increment the limit counter through field activation, much in the same way that a customer's product key is updated by the distributor. You may want to charge for incrementing the distributor's key. To increment the license limit counter in a distributor's key:

1. Start the SSP Toolkit and navigate to the **Implementation** stage.
2. Click the **Field Activation** tab, and then click **License Generator**.

3. Ask the distributor to generate a locking code for his distributor key using the Field Exchange Utility. The distributor key, not a product key, must be connected the distributor's system to generate the appropriate locking code.
4. Enter the locking code provided by the distributor, and select the **Increment Distributor Counter** action. You should have already programmed this action, and the increment value, when you initially created your protection strategy.
5. Generate a license code.
6. Send the license code to your distributor, who then enters the code in the Field Exchange Utility, automatically incrementing the license limit for his distributor key.

The distributor can now continue to activate and upgrade product keys for SceneryEditor, as described above.

### ***Choosing an Activation Type for Demo Applications***

When you want to provide your application as a demo, *you must use an activation type other than active*. This is because, by definition, a demo must expire (become inactive) after it is run a specific number of times. You will want to activate the demo when you ship it to your customers, so that it will run.

Therefore, you should use the **static** or **trusted** activation type. The demo is then active until it expires. If you want your distributor to activate the demo, you should use the **distributed** activation type.

---

**Note:** *The distributed activation type is similar to the trusted activation type; the only difference is that activations of distributed applications are metered.*

---

For more information on protecting demo applications, see “Controlling Demo Applications” on page 90.

### Network Licenses

Another decision you need to make while protecting your application is how you want to use *licenses* with your application. With SentinelSuperPro 6.1, every user of your application needs to obtain a license before he can run the application. The license allows the user to start the application and access the hardware key.

The *license limit* indicates the maximum number of concurrent users of the application. Each instance of an application uses a license when it is started.

Licenses can be used in two ways—with a *stand-alone* application or with a *network* application. If the application is stand-alone, each user needs his own hardware key, as only one license can be obtained from each key. If the application is a network application, only one key—located on the network—is required, but the single key can issue multiple licenses, allowing for simultaneous use of your application by several clients.

The type of licensing model to use is up to you. It depends on how you will be selling your application, and how you expect your users to deploy it within their organization.

### Sublicensing

Sublicensing is useful when you want to apply a license limit that is less than the key's factory-programmed (hard) limit. Typically, sublicenses are used only with network applications.

You can program up to 56 separate sublicense license limits in each key—each sublicense is a custom element occupying a single cell on the hardware key. The total number of sublicense limits you can program is dependent on the number of cells being used by other elements of your strategy.

In addition to defining your own license limits for the application as a whole, you can also use sublicenses to control concurrent access to specific features within a protected application. At runtime, access to a controlled feature is granted if the corresponding sublicense limit has not been exceeded.

---

## Getting Started

Now that you understand the concepts behind software security, you are ready to get started with designing your own protection strategy. The first decision you need to make is whether you want to use quick and easy protection, or if you want to take more time and create your own custom protection strategy.

### Quick and Easy Protection

If you don't have a lot of time and you need to quickly protect your software before shipping it to your customers, you may want to consider using automatic protection.

With SentinelSuperPro, implementing automatic protection for your application can take 30 minutes or less, while still giving you options for activation types, time and date controls and algorithm values. When you choose automatic protection, the SSP Toolkit does most of the work for you, with no need to modify your source code.

If you want to start protecting your applications right away, skip to page 95 to read important information about programming the hardware key, then go immediately to Chapter 6, "Starting the SentinelSuperPro Toolkit," on page 111 to get started.

### Customized Protection

If you have more time to work on your protection strategy, a custom strategy may be the answer for you. Customizing your strategy allows you to take advantage of a number of protection techniques, both basic and advanced, using those that work best for your application.

Customized protection allows you to:

- Choose how to use each memory cell in the key.
- Select the algorithms used in your protection strategy.
- Add data words or counters where appropriate.

- Add API function calls to your source code to support your protection strategy.
- Create a very secure level of protection for your application.

---

**Note:** *Creating your own customized protection scheme requires you to understand the API functions and all rules governing how cells can be programmed. Be sure you have thoroughly reviewed the information in this chapter, as well as Chapters 3 and 15, before you begin programming the key and writing code.*

---

If you decide you want to create your own unique protection strategy, continue with the next sections in this chapter, which provide information on the various techniques you can use to protect your application. Once you’ve completed this chapter, go to Chapter 6, “Starting the SentinelSuperPro Toolkit,” on page 111 to get started with implementing your strategy.

### Basic Protection Guidelines

If you decide to create your own customized protection strategy, keep in mind the following guidelines to ensure your strategy is effective.

#### ***Send Frequent Queries***

One of the most basic and effective techniques you can use to confuse hackers is to call the hardware key frequently. If you rely on a single call at the beginning of your code, it is relatively easy for a skilled hacker to isolate the call and defeat your protection.

Another potential problem with querying only once is that a user could remove the key after starting the application. The key could then be used to run another copy of the application. The first copy would continue to run, because no queries are being performed to check for the key’s continued presence. This process of removing a key after starting an application and then using the same key to start the application on other computers is known as “lamplighting.”

If you decide to implement network licensing as part of your protection strategy, you must send a message to the key every 90 seconds in order to maintain the license. Failure to send this “heartbeat” message to the server



(and thus the hardware key) will result in loss of the license and an error being sent to the application. Heartbeat messages let the server and key know that the license is still in use by the client running the application. For more information about heartbeat messages, see page 106.

### ***Scatter Lock Code***

Software locks consist of multiple steps: calling the key, evaluating the returned value, and acting on the evaluation results. For added protection, separate these lock components in your code. A software lock is harder to break if its code components are physically separated into different sections of the application instead of being located together.

### ***Manipulate Returned Data***

Use the data returned from the hardware key in various ways. For example, leave the result in a variable, then check it later.

---

## Commonly Used Protection Techniques

This section describes several common techniques you can use, individually or together, to protect your application. Many of these schemes are based on one or more of the following general methods:

- **Overloading** potential hackers with data by calling the hardware key many times throughout your code.
- **Decentralizing** your locks throughout the code, rather than restricting them to a few places that can be easily detected and eliminated.
- **Distracting** potential hackers with locks that make your application perform long series of meaningless operations. These calls mislead hackers and make your valid locks harder to isolate.

Some techniques can be used with returned values sent from any of the three type of words (data, counter and algorithms). Other techniques can be used with only one type of word.

---

**Note:** *A **returned value** is the value received from any type of cell in response to a query or read sent from your application.*

---

When evaluating a returned value, always compare the response to the **expected** value. Do not rely on receiving a specific invalid response.

### Reading Stored Data

For a simple protection scheme, program a single cell with a value. Then, have your application read that cell and verify it contains the correct data. If it does, continue execution. If the correct data is not found, assume the key is not attached or has been tampered with and proceed accordingly.

### Example

In this example, we programmed one cell in the key with a two-byte value. We then had our application read that cell during execution, taking appropriate action after the read.

1. Select a two-byte value. We used **1234**.
2. Select a cell to program this value into. We used cell **20**.
3. Use the SSP Toolkit to program the selected cell with the value. In this example, we programmed the value **1234** into cell **20**.

---

**Note:** To add a data word cell to your key, you need to create a custom element. See Chapter 8, “Working With Design Elements,” on page 169 for detailed instructions.

---

Use the **Data Word** cell type if you want the application to be able to modify the cell later. Use the **Locked Data Word** cell type if you want the cell to be read-only.

4. Add the required API functions in your application code:
  - **RNBOSproFormatPacket()** – Initializes the packet.
  - **RNBOSproInitialize()** – Performs required initialization.
  - **RNBOSproFindFirstUnit()** – Establishes communication with the key and gets a license.
  - **RNBOSproRead()** – Reads the cell and returns the value in it.
5. Code your application to evaluate the response to the read.
6. Code your application to display an error or abort if the read operation does not return the value you programmed. We recommend retrying the operation at least once before taking a negative action.

---

**Note:** For more information about the API functions used in this example, see Chapter 15, “API Function Reference,” on page 281.

---

## Using Algorithms to Encrypt Data

A more complex form of protection is using algorithms to encrypt data you send to the key.

In this case, you send a data string to the key that is encrypted by the key using a preprogrammed algorithm. Your application then examines the returned value, verifying that the correct encrypted string was returned, or using the value to control your application's execution in some way.

---

**Tip:** *Longer query strings generally offer greater protection. We recommend your query strings be at least 32 bits (8 hex characters) long.*

---

### Example

This example describes how to set up your application to require a correctly encrypted response from the key.

1. Select two 16-bit hex values to use for the algorithm. We used **1234** and **C000** to create an active algorithm using the enhanced algorithm engine.

Remember, the second word must be between 8000 and FFFF to make the algorithm active.

2. Select two cells to program these values in. We used cells **0A** and **0B**.

---

**Note:** *Throughout the SSP Toolkit, only valid and available cell addresses are provided in Address drop-down lists, preventing you from selecting an inappropriate address. For more information about algorithm address restrictions, see “Valid Algorithm Addresses” on page 47.*

---

SentinelSuperPro will select an address for you if you select **Auto** instead of a cell; see page 173 for more information.

3. Select a query string to send to the key to be encrypted. We used **8FA31B4B**.

4. In the SSP Toolkit, open the **Design** stage.
5. Add an **Algorithm** as a custom element. Enter the values you selected in step 1 as the first and second words of the algorithm.

See Chapter 8, “Working With Design Elements,” on page 169 for detailed instructions on adding algorithms to your protection strategy.

6. Move to the **Prototype** stage, and click **Go** to program a test key with your specifications.
7. Move to the **Implementation** stage, and click the **API Explorer** tab.
8. Click **Query Response Generator**.
9. On the Query Response Generator screen, determine the encrypted value your key will return for the query value you selected in step 3.

See “Querying Algorithms” on page 129 for more information on using the query response generator.

Using our examples from above, the query string **8FA31B4B** returns a value of **3C5E4AD1**.

10. Add the appropriate API functions in your application to make the query. The following functions are required:
  - **RNBOSproFormatPacket()** – Initializes the packet.
  - **RNBOSproInitialize()** – Performs required initialization.
  - **RNBOSproFindFirstUnit()** – Establishes communication with the key and gets a license.
  - **RNBOSproQuery()** – Sends the query string and points to a location for the response value.
11. Code your application to display a message and exit if the query does not return the appropriate response (determined in step 9).

## Using Activation Passwords

You can program the hardware key so an algorithm is associated with an activation password. The algorithm and the password are each two words long, and the password must immediately follow the algorithm. For example, cells 14 – 17 can be used as follows:

Algo (word 1)	Algo (word 2)	Activation Password (word 1)	Activation Password (word 2)
Cell 14	Cell 15	Cell 16	Cell 17

---

**Note:** See “Valid Algorithm Addresses” on page 47 for more information about address restrictions for algorithms and activation passwords.

---

By associating an activation password with an algorithm, you can activate the algorithm in the field. If an activation password is not associated with an algorithm, it cannot be activated in the field.

To set up your application to require an activation password:

1. Add an algorithm to your application via the SSP Toolkit, being sure to set it to inactive. See Chapter 8, “Working With Design Elements,” on page 169 for detailed instructions.
2. Code your application so that it executes only after receiving a valid response from the (currently inactive) algorithm.
3. Write a utility that uses the `RNBOSproActivate()` function to activate the algorithm once the user provides a password.
4. After buying your application, the user runs the utility you created in step 3, entering the password you provide. The algorithm is then activated and returns the correct response, allowing the protected application to execute.

For added security, you may want to use a different activation password for each key.

## Example

This example demonstrates how to activate an inactive application in the field. Usually, an application is deactivated because it is a demo that has been “turned off” after the specified number of executions. Alternatively, you may have set up your application so the user must enter an activation password before the application will run.

You temporarily “turn off” an application by including an RNBOspro-Query() function call that requires the hardware key to return a correctly encrypted value. Then you make it impossible for the hardware key to return the value because its algorithm has been set to inactive.

By definition, an algorithm is inactive if the high-order bit of its second word is 0. This is done as follows:

- If a counter is used, when it reaches zero, the RNBOsproDecrement() function sets the bit to 0.
- In your factory, the SSP Toolkit is used to set the algorithm as inactive.

**You must write a utility, or add a feature or function to your application, to activate the algorithm once the user supplies the correct password.** The query performed by the protected application then returns the correct response, and the application runs successfully.

---

**Note:** *The utility used to enter activation passwords is not included with, nor can it be created with, the SSP Toolkit. You must design and code this utility for use with your application yourself.*

---

The following example assumes you release your application in a deactivated state, and provide a password and utility to activate it.

1. Use the SSP Toolkit to program an algorithm with password as a custom element at address **0C**, as follows:
  - Algorithm: **0123 3456**
  - Password: **AB16 09C5**

Remember, the value in the second word of the algorithm must be between 0000 and 7FFF to make the algorithm inactive. See “Algorithm Values” on page 45 for detailed instructions.

2. Add API functions in your application to query the hardware key using the activated algorithm. The following functions are required:
  - **RNBOsproFormatPacket()** – Initializes the packet.
  - **RNBOsproInitialize()** – Performs required initialization.
  - **RNBOsproFindFirstUnit()** – Establishes communication with the key and gets a license.
  - **RNBOsproActivate()** – Passes the password input by the user, your write password, and the address of the algorithm’s first word.

If the password is correct, RNBOsproActivate() changes the algorithm’s active/inactive bit to active, making it available for queries.

---

**Note:** *You may want to send a query using the algorithm before calling RNBOsproActivate(). If the query returns the correct response, the algorithm is already activated.*

---

3. Write a utility the user can use to enter the password you provide. This utility should also use the API calls listed above.

### Dealing With Missing Hardware Keys

If no hardware key is attached to the computer or network server when a protected application is run, an error is returned by the RNBOsproFindFirstUnit() API function. If a connection is established, but the key is later removed, subsequent API functions will return errors. See “API Status Codes” on page 315 for more information about these errors.

If your application detects that the SentinelSuperPro key is not present, it is up to you to decide what action you want to take. Typically, you should not shut down your application because of a single unexpected response.



Instead, repeat your query; if the response is still wrong, then you can take action. Possible actions include:

- Display a message and wait for the user to respond. This method does not prevent users from running the application, but it makes doing so extremely annoying, especially if the application queries the hardware key frequently.
- Shut down the application after a predetermined number of failed queries. (However, only under the most extraordinary circumstances should you terminate your application without allowing the user to first save his work.)
- Allow the application to *appear* as if it is functioning properly, while in fact it is not. (Be very careful if you use this method; less drastic actions should be considered first.)
- Display a critical error message and tell the user to contact your technical support department.

These are just some suggested actions; you can implement any combination of them to suit your needs.

Remember, other events, such as network transmission errors or parallel port contention problems, can also cause your application to detect a hardware key problem. Since these are almost always innocent events, you should design your strategy to be as forgiving of them as possible, while still maintaining protection integrity.

---

**Note:** *All attempts have been made to guarantee error-free transmissions to and from the key. However, a small possibility exists that an invalid response may be received even if the key **is** attached. As a result, we recommend always retrying the query one or more times if you receive an invalid response. If the response is consistently invalid, then take the action you deem appropriate.*

---

## Dealing With Newly Connected Hardware Keys

Once a SentinelSuperPro server is running on your user's network, additional keys can be installed at any time without having to reboot the server—this is called *hotplugging*.

The SentinelSuperPro server will automatically detect any new keys that have been attached since the last time the server was started. This allows the user to connect a new key to the server without shutting down the server and terminating those clients currently accessing the key and running the protected application.

---

## Advanced Protection Techniques

Once you understand how to use the basic tenets of software protection, you may want to further protect your application through the use of advanced techniques.

### Using Returned Values as Variables

Because software is generally easier to break than hardware, most hackers will try to break your application by attacking the software. Therefore, any tricks or traps you can implement in your code by incorporating a response from the hardware key will add even more protection.

One effective technique is to hide software locks in a high-level language by using returned values to control application flow. With this method, a value returned by the key becomes a logical pointer or selection key to the next execution step or subroutine. This makes analysis of your code more difficult.

Another way to use a returned value is to add it to the value of a variable so the sum is the desired value of the variable. If the variable is used in other parts of the code, then that code is dependent on the call to the hardware key.

For example, suppose that at some point in your application you want a variable to contain the value 13. Assume that one of the query strings you send to the key returns the decimal number 12,345.

- Set the variable to -12,332.
- Send the query.
- Add the response to the variable.

If the correct key is attached, the variable will contain the proper value.

### Implementing Encryption Techniques

Another effective method for protecting your application is to use reversible encryption techniques to encrypt and decrypt data.

To do this, use the `RNBOsproQuery()` function to scramble a data string, and then use the scrambled response to encrypt your application code. You then ship your application to the field with encrypted code, which is decrypted only if the hardware key is attached.

Most encryption algorithms depend on a key value—sometimes called a password or seed—to transform the data.

Using a different seed produces different encrypted results, but reproduces the original data if that seed is also used for decryption.

---

**Note:** *Use of this technique requires advanced knowledge of encryption methods and their use in application code. Some of these techniques may be difficult or impossible to implement in some languages.*

---

### ***Using Returned Values as Encryption Seeds***

You can use the key's returned values to disguise critical portions of data or code as random data until decrypted for use. If the encryption seed is derived from values produced by SentinelSuperPro, the correct key must be present before the code can be decrypted and executed.

---

**Tip:** *When decrypted data is “in the clear,” use some other form of protection to block interrupts used by debuggers to gain control.*

---

The most common reversible algorithms use the Boolean operator EXCLUSIVE OR (XOR). XOR works as follows:

- If a seed bit has a value of 1, XOR reverses the state of the corresponding bit in the original string and copies it to the result.
- If a seed bit has a value of 0, XOR copies the corresponding bit in the original string to the result.

Applying the same algorithm to the result reverses the encryption and restores the data to its original state.

## Examples

The following example uses the XOR operator to encrypt a 16-bit hex number (8FA3) using the seed 4B6A.

Description	Hex	Binary
Data to encrypt	8FA3	1000111110100011
Seed	4B6A	0100101101101010
XOR algorithm		-----
Encrypted result	C4C9	1100010011001001

Notice that everywhere a bit in the seed is 1, the result bit is the opposite state of the data bit. Where the seed contains a 0, the result bit is the same as the original data bit. Without knowing the seed, the encrypted result is meaningless.

To reproduce the original data, apply the XOR algorithm to the encrypted result using the same seed.

Description	Hex	Binary
Encrypted result	C4C9	1100010011001001
Seed	4B6A	0100101101101010
XOR algorithm		-----
Original data	8FA3	1000111110100011

The next example shows how to use the SSP Toolkit with the XOR operator to encrypt and decrypt code in your application.

1. Select two 16-bit hex values to use for the algorithm. We used **4D59** and **F123**.

Remember, the second word must be between 8000 and FFFF to make the algorithm active.

2. Select two cells to program these values in. We used cells **0A** and **0B**.

SentinelSuperPro will select an address for you if you select **Auto** instead of a cell; see page 173 for more information.

---

**Note:** For more information about algorithm address restrictions, see “Valid Algorithm Addresses” on page 47.

---

3. Select query data to send to the key to be encrypted. We used **7009AB12**.
4. In the SSP Toolkit, open the **Design** stage.
5. Add an **Algorithm** as a custom element. Enter the values you selected in step 1 as the first and second words of the algorithm.

See Chapter 8, “Working With Design Elements,” on page 169 for detailed instructions on adding algorithms to your protection strategy.

6. Move to the **Prototype** stage, and click **Go** to program a test key with your specifications.
7. Move to the **Implementation** stage, and click the **API Explorer** tab.
8. Click **Query Response Generator**.
9. On the Query Response Generator screen, determine the encrypted value your key will return for the query data you selected in step 3.

See “Querying Algorithms” on page 129 for more information on using the query response generator.

The encrypted return value is the encryption seed you will use to encrypt part of your code. Using our examples from above, the query string **7009AB12** returns a value of **60D6867D**.

10. Select an encryption method. We used the Boolean operator XOR.
11. Select the data in your code you want to encrypt. We used the hex value **8FA31B4B**.

12. Apply the operator you selected in step 10 to the data, using the encryption seed. If you ship your application with the encrypted code, it will not execute correctly until the code is decrypted by a correct response.

In our example, the result is **A73FBA5B**.

13. Code your application so it decrypts the encrypted code if the hardware key is present. To do this, query the key. If it is present, the `RNBOsproQuery()` function returns the response string you used as the encryption seed.

Because XOR is a reversible operation, applying the same encryption seed to the encrypted data returns the data to its original state and your application should continue to execute properly.

14. Add the following required API functions in your application to make the query:

- **RNBOsproFormatPacket()** – Initializes the packet.
- **RNBOsproInitialize()** – Performs required initialization.
- **RNBOsproFindFirstUnit()** – Establishes communication with the key and gets a license.
- **RNBOsproQuery()** – Sends the query string and points to a location for the response value.

Code your application to display a message and exit if the query does not return the appropriate value and the code cannot be decrypted.

### ***Using Longer Encryption Seeds***

If the data to encrypt is longer, a longer seed can be constructed. The scheme for forming such a seed may be as complicated as you wish.

For example, the number 4B6A can be expanded to a 32-byte string by “rotating” it left 15 times and stringing the results of each rotation together. This yields the following hex string:

```
4B6A 96D4 2DA9 5B52 B6A4 6D49 DA92 B525  
6A4B D496 A92D 525B A4B6 496D 92DA 25B5
```

You can use this string as a seed with the XOR algorithm to encrypt a 32-byte string. For example, the ASCII string “This is the secret of my program” can be represented as the following hex string:

```
5468 6973 2069 7320 7468 6520 7365 6372  
6574 206F 6620 4D59 2070 726F 6772 616D
```

Using the 32-byte seed with the XOR algorithm produces the following encrypted result:

```
1F02 FFA7 0DC0 2872 C2CC 0869 A9F7 D657  
0F3F F4F9 CD0D 1F02 84C6 3B04 F5A8 44D8
```

The result looks nothing like the original character string, yet the original data can be easily recovered using the same algorithm seed that changed it.

You can use this method with entire sections of code within your application, expanding the seed as needed.

### ***Using Advanced Encryption Techniques***

You can make encryption even more complex, depending on how sophisticated you want to make your application. For example:

- Use values returned by the key as seeds for a pseudo-random number generator that generates seed encryption patterns.
- Use returned values to decrypt subroutines that then decrypt code using an entirely different encryption method and seed.
- Instead of using the XOR operator, multiply each byte by a seed to encrypt it. Divide by the same seed to decrypt the data.

Multiplying an 8-bit value can yield a 16-bit result. The result is double the size of a data string produced with the XOR operator, but is also harder to crack. If you use this technique, make sure your multiplier/divisor seed does not equal 0.



Your local technical library should have several reference materials on encryption that can help you implement these techniques in your protection strategy. Other topics you may wish to research include codes, cryptology and the National Security Agency (NSA).

## Querying Activation Passwords

Normally, an activation password is used to activate an inactive algorithm, as described in “Using Activation Passwords” on page 76. However, because an algorithm password has an access code of 3 (meaning it is also an algorithm word), you can also use the password itself as an algorithm.

To make a password an active algorithm, you must set bit 7 of its second word to 1 (set the value in the second word to a number between 8000 and FFFF). Then use the `RNBOSproQuery()` function to send an input string to the key, specifying the starting address of the password. The key encrypts the string according to the bit pattern of the algorithm password, and then returns the encrypted value to your application.

This technique provides an alternate method of querying the key. For example, you may want to query the algorithm password before invoking the `RNBOSproActivate()` function, to verify that the password appears to be correct.

---

**Note:** *You cannot use this method if you use the trusted activation type, which creates a different activation password for each customer.*

---

## Using Data Words

In addition to reading the value in a single data word cell (see page 72), there are a number of other ways you can use cells programmed with read/write data words (access code 0) or read-only data words (access code 1) as part of your protection strategy.

- Store machine code in data words. This code can be read, checksummed and executed in a way that is verified by a different part of the application.

- Program the application's serial number in a data word cell. Read the cell and compare the value to the correct serial number.

If you have multiple application packages, store the serial number for each in separate data word cells.

- Store the user's name in data words as ASCII bytes, then compare or display it.
- Use the 56 programmable cells as one large, 896-bit bitmap. Various combinations of bits can determine features or other responses, depending on your application.

### **Assembly Language Techniques**

Implementing SentinelSuperPro protection in assembly language offers more flexibility than other languages. However, you can use only one SentinelSuperPro subroutine to make hardware key queries. If you try to link two different interface subroutines with your application, you may get doubly defined symbols.

#### ***Hiding Calls***

A hacker may analyze your object code and examine addresses referenced by CALL instructions to find the calls to the SentinelSuperPro interface routines. The hacker could then analyze the code of the interface routine and the code following each call in order to defeat the lock.

One method to avoid detection of your queries is to call the key without using the assembly language CALL instruction. Instead, push the return address onto the stack followed by the procedure address, and then execute a RET (return) instruction.

#### ***Inserting Extra Data***

Analysis of your code can also be made more difficult by inserting frequent “garbage” data bytes. This process is effective at throwing static disassemblers out of sync.

For example, after each unconditional jump and return, insert a garbage data byte or two whose value is equal to the first byte of a very long assembly language instruction.

This same technique can be used following conditional branches, as long as the preceding code always guarantees the branch is invoked. Such a jump or branch may also be used immediately prior to the call, with an intervening data byte.

## Using Stepped Access

If you market multiple versions of your application, you can use SentinelSuperPro to control access to features within the application, based on criteria you specify. This is called *stepped access*.

For example, you may offer a basic package, an expanded package with some additional features, and a deluxe package with all features. Using stepped access, the application contains an array of conditions instructing the system to activate different features based on the value returned by the hardware key. You control the features implemented by the application by using a different algorithm for each package.

Three algorithms produce three different values for the same string. For the string ABCD, for example, three algorithms might produce the values 2610, 1830 and 6287.

Your application should contain statements that produce different responses based on the returned value, as illustrated by the following pseudocode:

```
If <result> EQUALS
2610 THEN <enable basic features>
1830 THEN <enable basic + expanded features>
6287 THEN <enable all features>
```

If you have many steps, or conditions, they can be stored in an array. The application checks the array for a match with the string and returns the number of the element matched. This number then determines the features activated or the action taken by the application.

## Obstructing Debuggers

Many potential hackers use debuggers to break large, complex software packages with high licensing fees. You may want to incorporate safeguards aimed directly at preventing the use of debuggers to circumvent the software locks in your protection strategy.

For example, you might lock out the keyboard during hardware key queries, or destroy the contents of interrupt vectors 1 and 3 (the trace and break-point interrupts).

While no technique can deter every hacker, the more safeguards you implement, and the greater the variety you use, the more difficult the hacker's task. Eventually, it makes more sense for potential hackers to either purchase your application, or attempt to break a different, less secure application.

---

## Controlling Demo Applications

If you are shipping a demo version of your application, you may want to control the number of times the application is executed, and then prevent its use after a predetermined number of executions. You can use SentinelSuperPro to count the number of times the application is executed.

To count executions, you program a cell as a counter, and set its initial value. Each time the application is executed, the counter is decremented by one (the decrement is performed by the RNBOsproDecrement() function). You code your application to not run again once the counter reaches zero.

If you use the shelled protection type, you can also limit demo applications by time and number of days. For example, your application may be enabled for only 30 days after the initial execution. For more detailed information on using time and date controls see page 156.

The counter/algorithm combination may also be associated with a password. If the user purchases an extension of the software, you provide them with the activation password that gives them full, unlimited access to your software. See page 76 for more information about providing users with activation passwords in the field.

Like the full version of an application, demos also require a license to access the hardware key. You may want to consider whether you want your demo applications to act as stand-alone applications or as network applications. Both demo applications and the full version can be run on the network at the same time, as long as enough licenses are available. For more information about using licensing, see “Network Licenses” on page 68.

## Using Counters

There are multiple ways of using counters to control access to your demo applications. One way is to check the counter to see if it has reached zero, then proceed accordingly.

Another, more secure, method is to associate the counter with an algorithm the application requires for queries. When the counter reaches zero, the associated algorithm is automatically deactivated. (RNBOsproDecrement() changes the high-order bit of the algorithm’s second word.) Future queries using this algorithm return incorrect values.

To limit application executions using an algorithm/counter, you must program the key to meet the following specifications:

- The word you decrement must be a counter word.
- The counter must be located in a cell with an address equal to  $3 \text{ MOD } 8$ .
- The two words immediately following the counter (at addresses equal to  $4 \text{ MOD } 8$  and  $5 \text{ MOD } 8$ ) must contain an active algorithm.

---

**Note:** *The relationship between a counter word and an adjacent algorithm exists even if you do not intentionally plan it. The algorithm will be deactivated when the counter reaches zero. See “Valid Algorithm Addresses” on page 47 for more information about where counters can be placed.*

---

You can also use two counters: the algorithm is deactivated when either counter reaches 0. The second counter must be located at an address equal to  $2 \text{ MOD } 8$ . If desired, you could use the second counter after the algorithm is re-activated with a password.

If you want to be able to re-activate the application after it has been disabled, you must define an activation password. This is a two-word value immediately following the algorithm. See “Using Activation Passwords” on page 76 for more information.

Remember, the counter will still be zero after the algorithm is re-activated, so make sure your application checks for the ALREADY\_ZERO status from the key.

---

**Note:** *You could reset the counter and do decrements again, but this would require putting your overwrite passwords in your activation utility. You should avoid using your overwrite passwords in the field.*

---

Any of the following cell type groups can be used to program an algorithm with a counter:

- Algorithm with counter
- Algorithm with counter and password
- Algorithm with two counters
- Algorithm with two counters and password

The following illustrates a cell layout with an algorithm, counter and password:

Counter (CA)	Active Algorithm (AA)	Active Algorithm (AA)	Activation Password (AP)	Activation Password (AP)
Cell 0B	Cell 0C	Cell 0D	Cell 0E	Cell 0F

Cell 0B contains the counter you are decrementing. Cells 0C and 0D contain the active algorithm. The second algorithm word must be between 8000 and FFFF for the algorithm to be active—this value will be changed automatically when the counter reaches zero.

Cells 0E and 0F contain the activation password required to re-activate the password after the counter reaches zero to deactivate the algorithm.

## Example

This example demonstrates how to limit the number of times a demo application can execute. This is done by requiring the application to query an algorithm that becomes unusable (deactivated) after a specified number of executions. The algorithm is associated with a counter that is initialized to the number of times the application can run.

**Remember, if you want to allow the user to re-activate the application after the counter has reached zero, you must also program an activation password.**

---

**Note:** *This example assumes you will be controlling the number of times your demo can execute by adding a custom element. You also have control over demo executions when you use integrated or automatic application protection. See Chapter 7, “Protecting Your Application,” on page 141 for more information.*

---

1. Select the cells you want to use for the algorithm and counter. Review the address restrictions on page 47.

We used cells at the following addresses:

- Counter: **0B**
- Algorithm Word 1: **0C**
- Algorithm Word 2: **0D**

2. Decide how many times you want the demo to run. We chose **5**.
3. Select two 16-bit hex values to use for the algorithm. Remember, the second word must be between 8000 and FFFF to make the algorithm active.

We used **1234** and **C000**.

4. Select an input string (preferably at least 32 bits long) to send to the key. We used **ABCDDCBA**.

5. Use the SSP Toolkit to program an algorithm with counter as a custom element at the address you selected for the counter cell. See Chapter 8, “Working With Design Elements,” on page 169 for instructions.

We programmed it at address **0B**, as follows:

- Algorithm: **0123 C000**
- Counter: **5**

---

**Note:** *If you want to provide the ability to re-activate the application in the field, you must add an algorithm with counter and password instead. This allows you to program an activation password into the two cells following the algorithm.*

---

6. Add the following required API functions in your application:

- **RNBOsproFormatPacket()** – Initializes the packet.
- **RNBOsproInitialize()** – Performs required initialization.
- **RNBOsproFindFirstUnit()** – Establishes communication with the key and gets a license.
- **RNBOsproDecrement()** – Decrements the counter by one. Call this function every time your application executes.

In our example, on the fifth execution, the counter in cell 0B reaches zero. At this point, RNBOsproDecrement() deactivates the algorithm used by the RNBOsproQuery() function (see below). On the sixth execution, therefore, the query does not return the expected value. Usually, you would code the application to display a message and then terminate.

- **RNBOsproQuery()** – Sends the query string and points to a location for the response value.

---

**Tip:** *To make a potential hacker’s task more difficult, separate the RNBOsproQuery() and RNBOsproDecrement() function calls in your code. This helps obscure the connection between them.*

---



## Querying Counters

If you are using a counter with your demo application, you can use the `RNBOsproQuery()` function to query the counter word, verifying it has been counted down. This technique is useful because it allows you to see if a hacker is trying to sidestep your counter in an attempt to continue the demo condition indefinitely.

Counters used in this way must be two words long:

- The first word has an access code of 2 (counter). This word must be located in an even-numbered cell.
- The second word has an access code of 3 (algorithm/hidden). This word must be located in an odd-numbered cell. The value of this word **must** be in the range for active algorithms.

Code your application to send a query to the counter/algorithm before performing each decrement. A given query string should return a *different* response value after each decrement, because the value in the first word (the counter) will have changed.

If the query returns the *same* value, either the algorithm is not active, or the counter was not decremented, which may mean a hacker is attempting to circumvent your protection strategy.

---

## Programming the Hardware Key

Once you have an idea of the types of protection you want to use, you must consider how you want to program the 56 available cells on the hardware key to implement your strategy.

A primary consideration is the number of applications that will share the hardware key. Any of your applications can use any available cell, but you must make sure to allow enough cells for the type of protection you want for each application.

**Note:** See Chapter 3, “Using the Hardware Key,” on page 33 for more information about the hardware key and the available cell types.

Using One Key for Multiple Applications

One SentinelSuperPro key can be used to protect multiple applications, up to 28 applications per key, depending on your protection strategies.

Usually, you protect multiple applications on one key by designating certain cells for each application.

The sample layout below illustrates how you can use one key to protect seven applications. Each application is assigned a group of cells consisting of two data words (cell type DW), two algorithm counters (cell type CA), one inactive algorithm (cell type IA), and an activation password (cell type AP).

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
00	SN	DI	RW	RW	RW	RW	RW	RW	
08	DW	DW	CA	CA	IA	IA	AP	AP	App 1
10	DW	DW	CA	CA	IA	IA	AP	AP	App 2
18	DW	DW	CA	CA	IA	IA	AP	AP	App 3
20	DW	DW	CA	CA	IA	IA	AP	AP	App 4
28	DW	DW	CA	CA	IA	IA	AP	AP	App 5
30	DW	DW	CA	CA	IA	IA	AP	AP	App 6
38	DW	DW	CA	CA	IA	IA	AP	AP	App 7

Sample Key Programmed to Protect Seven Applications

Application 1 uses cells 08–0F, application 2 uses cells 10–17, and so on.

---

## Moving On

Now that you have designed your protection strategy, you are ready to use the SSP Toolkit to add software locks to your code and program your hardware keys. Go to the next chapter for instructions on starting the SSP Toolkit.



# Chapter 5

---

## Implementing Licensing

In SentinelSuperPro 6.1, all protected applications must obtain a license before they can be run. Licenses determine both who can use the application, and where the application can be used. Each instance of an application uses a license when it is started. As a developer, you have control over where your application will look for a key and obtain a license.

SentinelSuperPro functions you will add to your application's source code will locate a key, obtain a license, maintain the license by sending messages back to the server, and then release the license when it is no longer needed.

This chapter explains how licensing is implemented in your protected applications. Detailed instructions for adding the appropriate code can be found in “Adding API Functions to Your Source Code” on page 192.

This chapter covers the following topics:

- Setting the access mode
- Finding a key
- Getting a license
- Maintaining a license
- Releasing a license
- Using sublicenses

---

## Setting the Access Mode

Access modes determine where your application will look for the appropriate hardware key. There are three access modes that can be used by your protected application: *stand-alone*, *network* and *dual*.

- **Stand-alone mode** is used for applications where you want *only* a local key to be used.
- **Network mode** is used for applications where you want *only* a network key to be used.
- **Dual mode** is used when you want your application to use either a local key or a network key. This is the default mode for all SentinelSuperPro-protected applications.

Depending on how you want your protected application to be used, you can allow your users to set the access mode through an environment variable, or you can set the access mode through code. An access mode set through code will always override an access mode set through an environment variable.

---

**Note:** *If you don't specify an access mode, your protected applications will use dual mode as the default.*

---

### Setting Stand-alone or Network Mode

Stand-alone or network mode is set through the **RNBOsproSetContact-Server** API function. This function tells the application whether or not to look on the network for a server and key. If the function is set to “**no-net**”, the application will look only on the local system for a key. If the function is set to a specific server name, the application will look only to that server for a key.

Putting your application in network mode by setting the contact server prevents the application from sending a broadcast message to the entire network.

Sending broadcast messages not only requires use of network resources, it also takes longer, meaning the application will take longer to load and be ready for use. Setting your application to network mode will save both time and network resources, because the application will contact *only* the server you have defined in the `RNBOSproSetContactServer` API function or the `NSP_HOST` variable.

---

**Note:** *If you set your application to network mode in code, your user must name his SentinelSuperPro server the same as the server name you set. If he does not, your application will be unable to locate the server and key, and will not run. For this reason, if you want your application to run in network mode, you may find it easier to use dual mode in code, and then instruct your users to set the server name through the `NSP_HOST` variable. You could also use a configuration file to set the variable. See page 102 for more information about this variable.*

---

To set stand-alone mode:

- `RNBOSproSetContactServer(apiPacket, "no-net");`

To set network mode:

- `RNBOSproSetContactServer(apiPacket, "**host name/  
IPaddress/IPXaddress of serverhost*");`

For specific details about using this function, refer to Chapter 15, “API Function Reference,” on page 281.

## Setting Dual Mode

Because dual mode is the default access mode, you do not need to add any special API functions to your source code. The standard pseudocode (see “Adding API Functions to Your Source Code” on page 192) assumes your application will be used in dual mode.

When in dual mode, an application will send broadcast messages to the network to locate an appropriate server. Keep in mind, broadcast messages require additional network resources and result in a longer total time from application start-up to key acquisition. If network resources and timing is an issue for you, you may want to consider using network mode.

## About the NSP\_HOST Variable

Another means of setting the application's access mode is through the NSP\_HOST environment variable. NSP\_HOST tells the protected application which server to look for when it needs to access a SentinelSuperPro key.

This variable is typically set by your end-user on the server or client systems your application will be run on. However, in all cases, *an access mode set through code will override an access mode set via the environment variable*. We recommend setting the access mode through code whenever possible.

The possible values for this variable are as follows:

- **no-net:** Tells the application to act as a stand-alone application. The application will look for a key *only* on the client machine.

If the key is not found, the application will *not* send a broadcast message to the network looking for a server and key.

- **server host name, IP address or IPX address:** Tells the application to act as a network application. The application will look for a key *only* on the selected server.

If the selected server is not found, or a key is not found on the selected server, the application will *not* send a broadcast message to the network looking for another server and key.

Setting this variable is optional, and, if you have set the access mode through code, unnecessary. However, for maximum performance, we recommend to system administrators that this variable be set on each client workstation. If the end-user does not set this variable, and you have not set the access mode through code, the application will be in dual mode.

Detailed instructions for setting this variable can be found in the *SentinelSuperPro System Administrator's Guide*.



---

## Finding a Key

Before your application can obtain a license, it must first locate a Sentinel-SuperPro hardware key either on the local machine or somewhere on the network. Where and how your application locates a key is dependent on what access mode your application is using.

The following API function calls are used to locate a key:

- RNBOsproFormatPacket
- RNBOsproInitialize
- RNBOsproGetContactServer (optional)
- RNBOsproFindFirstUnit

### Finding a Key in Stand-alone Mode

When your application is in stand-alone mode, it looks for a key on the local machine. If a key is not present locally, the application will not go to the network to find a key. It will simply return an error message that a key cannot be found.

If the key is found, the application starts normally and communicates with the key through the server (which must also be installed on the same workstation) to obtain a license.

### Finding a Key in Network Mode

When your application is in network mode, it will go immediately to the network to locate a key, ignoring the local USB and parallel ports completely.

If a contact server was set in code using the RNBOsproGetContactServer function, or through the NSP\_HOST variable, the application will look only for the specified server. If the server cannot be located, an error message appears and the application shuts down.

The application does not look for a key locally when it is in network mode.

## **Finding a Key in Dual Mode**

While in dual mode, the application will always check for the presence of a key on the local machine before it goes out to the network. If a local key is found, the key is used as if in stand-alone mode.

If the application cannot find a local key, it sends a broadcast message to the network to find out which servers are available. It then communicates with one of those servers to obtain a license from the key. If a network key is found, the key is used as if in network mode.

### ***Finding the First Server by Sending a Broadcast Message to the Network***

An application running in dual mode will send a broadcast message to the network to locate a server and key only when a local key cannot be found. The broadcast message is sent to the subnet the application is running on. Once the first server responds, the application attempts to obtain a license from that server.

If the contacted server does not have any licenses available, no further broadcast messages will be sent, and the application will return an error.

### ***Finding Additional Servers***

To allow your application to obtain a list of all servers available on the network, use the `RNBOSproEnumServer` API function.

This function allows you to obtain the server list, and then contact each server on the list until a license is obtained. The following functions must be called prior to contacting each server: `RNBOSproSetContactServer` and `RNBOSproFindFirstUnit`. See page 290 for more information about the `RNBOSproEnumServer` function.

---

## Getting a License

All license information is maintained by SentinelSuperPro server software installed on the client machines or the servers on the network.

Before an application can be started, it must first obtain a license from the SentinelSuperPro server. The server issues a license only if the *license limit* in the key has not yet been exceeded. The license limit indicates the maximum number of concurrent users of the application. Once a key is located, the way in which a license is obtained is the same for applications running all modes.

---

**Note:** *Remember, whether the application is stand-alone or network, it always uses the SentinelSuperPro server as the means of communicating with the key. This is why the SentinelSuperPro server must be installed on the same system as stand-alone applications, and on a server located on the network for network applications.*

---

The process for obtaining a license is as follows:

1. Once a hardware key is found, the application requests a license.
2. The SentinelSuperPro server queries the Sentinel driver to obtain the license limit from the hardware key.
3. The driver returns the license limit to the SentinelSuperPro server.
4. If a license is available, the SentinelSuperPro server grants the license. If all licenses are in use, the server denies the license request.
5. The SentinelSuperPro server communicates the license status—granted or denied—to the client.
6. One of the following occurs:
  - If the license was granted, the application continues to run.
  - If the license was denied, the user is notified that no licenses are available. You can choose to then display an error or shut down the application.

---

**Note:** *All communication between the client and the server is encrypted for greater security.*

---

### The License ID

Once a license is obtained, a *license ID* that uniquely identifies the client and the license is issued. The license ID number must be used in all subsequent calls to the key from the client.

### Maintaining the License

Once a license has been obtained, the application must maintain the license by sending “heartbeat” messages to the server, confirming the client is still using the license. To send a heartbeat message, call any API function that accesses the key, such as a read or write function. When these functions access the key, they confirm to the server that the key (and license) is still in use.

These periodic messages should be sent every 90 seconds. If the server does not receive a heartbeat message from the client, it will release the license and send an error to the application. An error is also returned if the application sends a heartbeat message after the license has already been released. How to handle these errors in your application is left up to your discretion.

### A Note About Licenses

Throughout this manual, we refer to *obtaining* and *releasing* licenses. Although it is convenient to describe license management in this way, in actuality, licenses are never physically moved between the server/key and the client workstation. Instead, the SentinelSuperPro server simply keeps track of how many users can run the application and decrements and increments the license count as authorized users are granted permission to run the application and as they exit the application.

---

**Note:** *The SentinelSuperPro server maintains a log of all transactions that take place during a particular session, allowing you to view when a license is issued and who it was issued to. See the SentinelSuperPro System Administrator’s Guide for more information about the server log file.*

---

---

## Releasing a License

There are three situations in which a license should be released:

- The client has shut down the protected application
- The client fails to send a heartbeat message to the server
- Your application has completed all key operations

Use the `RNBOsproReleaseLicense` API function to release a license and make it available for use by other clients. For more information about using this function, see page 311.

---

## Using Sublicenses

A sublicense is a license limit you define that is less than or equal to the hard limit programmed into the key. Sublicenses allow you to:

- Implement fewer licenses for an application than the hard limit programmed on the key
- Protect several applications using the same key, and define separate license limits for each
- Control concurrent access to specific features or modules within your protected application(s)

Sublicenses must be a constant number. But, there must be a hard license available before a sublicense can be obtained, no matter how many sublicenses are left.

## Sublicense Usage Example

For example, assume the hard limit on your network keys is 20 licenses. If you are protecting three applications (SceneryEditor, ShapeEditor and TextEditor) with a single key, you could use sublicenses to define the following:

- Set the license limit for SceneryEditor to 10
- Set the license limit for ShapeEditor to 7
- Set the license limit for TextEditor to 10

Notice that the total number of sublicenses is greater than the hard limit. This means that if all the sublicenses for SceneryEditor are being used, there are only 10 hard licenses left.

If all seven licenses are being used for ShapeEditor, only three hard licenses are left for TextEditor. Thus, even though TextEditor has 10 sublicenses available, only three clients can use it. This is because the hard license is obtained first, then the sublicense.

## Getting a Sublicense

To obtain a sublicense for a client, first use the `RNBOSproFindFirstUnit` function to obtain a license from the key, then use the `RNBOSproGetSubLicense` API function to obtain the sublicense.

The key's hard limit is decremented first, then the sublicense limit is decremented for the requested application. It is up to you as to how you want to handle a client request when a sublicense is unavailable.

We recommend that if the sublicenses for a particular application are all being used, no additional clients should be allowed to obtain a license for that application, even if there are still licenses available under the hard limit.

Once the sublicense has been obtained, it works in the same way as other licenses—you can read or write to cells on the key, activate algorithms and more. You also need to send heartbeat messages to maintain the sublicense in the same way that you would to maintain a normal license.

## Adding Sublicenses to Your Protection Strategy

Sublicenses are added to your protection strategy in the same way that counters and data words are added—as a custom element. For more information about adding sublicenses, see “Adding Sublicense Limits” on page 182.

Because sublicenses are stored in locked data word cells, you can have as many sublicenses as you have available cells. This gives you flexibility in implementing sublicenses in your protection strategy. For example, you could use a sublicense for accessing a particular algorithm—the algorithm could only be accessed if the sublicense limit has not been exceeded.

Or, you could use a single sublicense to limit usage of two separate algorithms by programming your application—using the `RNBOSproGetSublicense` API function—to get the sublicense limit (located in a single cell) prior to querying either algorithm. If the sublicense limit has been exceeded, either algorithm cannot be successfully queried.

---

**Note:** *Cell availability depends on the number and type of elements being used in your protection strategy, as well as the number of applications being protected.*

---





# Chapter 6

---

## Starting the SentinelSuperPro Toolkit

The SentinelSuperPro Developer's Toolkit features an intuitive interface which allows you to start protecting your applications immediately. This chapter provides a guided tour of the SSP Toolkit, introducing ways to maneuver smoothly from stage to stage, as well as instructions for viewing, entering and modifying information. We suggest reading this chapter to get a feel for how the SSP Toolkit works before performing any protection tasks.

This chapter covers the following topics:

- Opening and navigating in the SSP Toolkit
- Learning about SentinelSuperPro functionality
- Using the API Explorer
- Creating a new project or opening an existing project
- Creating a project file for distributors
- Saving and locking your project

---

**Note:** The procedures in this chapter assume you have already installed the SSP Toolkit and the SentinelSuperPro Server. If you have not, please refer to Chapter 2, “Installation,” on page 19 to install the software, then return to this chapter once it is installed.

---

---

## Opening the SentinelSuperPro Toolkit

The SentinelSuperPro Server must be running in order for the Toolkit to be able to access the hardware key while you create your protection strategy. Therefore, before starting the Toolkit, verify you have the server running on your system. See the *SentinelSuperPro System Administrator's Guide* for more information about using the server.

To open the SSP Toolkit:

1. From the **Start** menu, point to **Programs > Rainbow Technologies > SuperPro > 6.1**.
2. Select **SuperPro 6.1 Toolkit**. The Developer Configuration dialog box appears.

### Entering Your Passwords

Before you can use SentinelSuperPro, you must provide your **developer ID**, **overwrite passwords**, and **write password**.

---

**Tip:** *You only need to enter these the first time you open SentinelSuperPro, as your passwords are remembered for subsequent sessions.*

---

Your *developer ID* is a unique identification code. You must use your developer ID to program or establish a connection to your keys. All the keys used by your organization have the same developer ID.

The *overwrite passwords* allow you to set or change the value or access code of **any** cell other than a restricted cell. Keep these passwords secure, as they have the power to reprogram all other cells in your key!

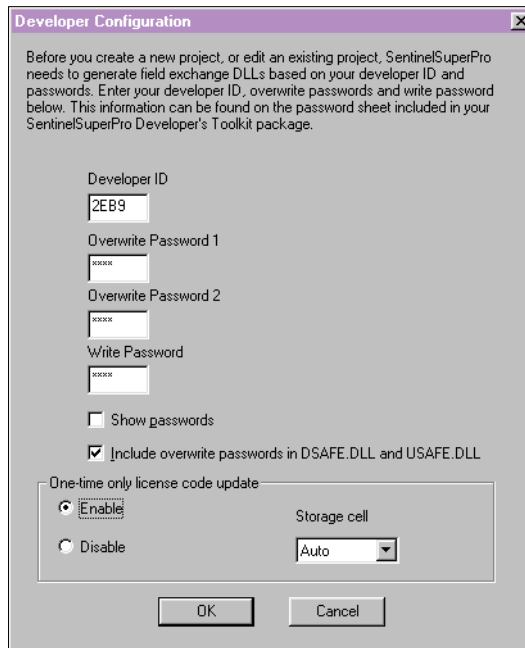
The *write password* allows you to change or set the value or access code of a data word or undefined cell. This password also allows you to decrement counter words.

These passwords, and your developer ID, are provided by Rainbow Technologies, and can be found on the password sheet included in your SentinelSuperPro package.

Once you enter the developer ID and passwords, the field exchange DLLs are created. These DLLs use the developer ID and passwords for the attached key. These DLLs are used to activate or update keys in the field.

If you enter an incorrect developer ID or passwords, you will be unable to program keys in the Prototype stage, and thus will be unable to implement your protection strategy.

To enter your passwords:



The image shows a 'Developer Configuration' dialog box with a purple title bar. It contains a text area with instructions, several input fields for Developer ID, Overwrite Password 1, Overwrite Password 2, and Write Password, a checkbox for 'Show passwords', a checked checkbox for 'Include overwrite passwords in DSAFE.DLL and USAFE.DLL', and a section for 'One-time only license code update' with radio buttons for 'Enable' and 'Disable', and a 'Storage cell' dropdown menu set to 'Auto'. At the bottom are 'OK' and 'Cancel' buttons.

**Developer Configuration**

Before you create a new project, or edit an existing project, SentinelSuperPro needs to generate field exchange DLLs based on your developer ID and passwords. Enter your developer ID, overwrite passwords and write password below. This information can be found on the password sheet included in your SentinelSuperPro Developer's Toolkit package.

Developer ID  
2EB9

Overwrite Password 1  
XXXX

Overwrite Password 2  
XXXX

Write Password  
XXXX

☐ Show passwords

☒ Include overwrite passwords in DSAFE.DLL and USAFE.DLL

One-time only license code update

☒ Enable ☐ Disable

Storage cell  
Auto

OK Cancel

**Developer Configuration Dialog Box**

1. In the **Developer ID** field, enter your developer ID.

---

**Tip:** *If you want to be able to see the actual password and developer ID characters, select the **Show Passwords** check box. Password characters are hidden (displayed as asterisks) by default.*

---

2. In the **Overwrite Password 1** field, enter the first overwrite password for your key.
3. In the **Overwrite Password 2** field, enter the second overwrite password for your key.
4. In the **Write** field, enter the write password for your key.
5. If you want to include the overwrite passwords in the field exchange DLLs, or use the one-time update feature, go to “Including Overwrite Passwords in the Field Exchange DLLs” on page 115. Otherwise, go to the next step.
6. Click **OK**.

If you have already created projects with the Toolkit, be sure to read the warning message that appears thoroughly and take the appropriate action.

The Toolkit window appears.

7. Go to “Navigating in the SentinelSuperPro Toolkit” on page 117.

### **About the Field Exchange DLLs**

The field exchange DLLs—*dsafe32.dll* and *usafe32.dll*—are generated each time you enter or change the developer ID and/or passwords in the Developer Configuration dialog box. If DLLs already exist on this workstation, they will be overwritten, as each time DLLs are generated they are different.

If new DLLs are generated, and you have already distributed the old DLLs with your protected software, you will be unable to reprogram keys using the old DLLs because your *dsafe32.dll* won't match with the user's *usafe32.dll*. In this case, be sure to create backups of your old DLLs so you can continue to reprogram existing keys in the field.

## Including Overwrite Passwords in the Field Exchange DLLs

The overwrite passwords are necessary whenever you want to change the value of a locked data word or read-only cell on the key. If you will be implementing the one-time update option for license codes (see below), you must include the overwrite passwords in your field exchange DLLs.

When included in the DLLs, the overwrite passwords are encrypted. However, **Rainbow Technologies strongly recommends against including these passwords in your field exchange DLLs**, as they could pose a security risk to your application. A talented hacker could possibly decrypt these passwords, and use them to gain unauthorized access to your application. Think carefully before you choose to include the overwrite passwords in the field exchange DLLs.

To include the overwrite passwords in the field exchange DLLs:

1. Select the **Include overwrite passwords in DSAFE.DLL and USAFE.DLL** check box. A warning message appears.
2. Read the warning message, then click **OK**.
3. Go to the next section.

## Enabling the One-Time Update Option for License Codes

The one-time update option allows you to prevent license codes from being applied to a key more than once. The license code includes information about how a key should be updated, such as incrementing a license counter or activating an application. (For more information about license codes, see “What Is a License Code?” on page 246.)

For example, if a single license code that increments a counter is applied to a key multiple times, the counter will be incremented multiple times also. This means that if a license code that increments a counter by five is applied three times, the total incremented value will be 15—10 more than you intended.

To prevent users from applying a license code more than once, enable the one-time update feature. This feature uses a single cell to determine the validity of a license code. If the value in the cell is not what is expected by the

license code, the license code cannot be applied, and the key will not be updated.

---

**Note:** *The one-time update option is available for use with product keys only. It is not implemented for distributor keys.*

---

The one-time update feature requires the overwrite passwords to be included in your field exchange DLLs (see page 115). You may want to weigh the benefits of using the one-time update feature against the security risks inherent in including the overwrite passwords in the DLLs.

To enable the one-time update option:

1. Verify that the **Include overwrite passwords in DSAFE.DLL and USAFE.DLL** check box has been selected.
2. Under **One-time only license code update**, select **Enable**. A warning message appears.
3. Read the warning message, then click **OK**.
4. In the **Storage Cell** field, select the cell you want the one-time update feature stored in. Select Auto to allow SentinelSuperPro to select a cell for you.

Once you select this cell, it cannot be moved or changed.

5. Click **OK**.

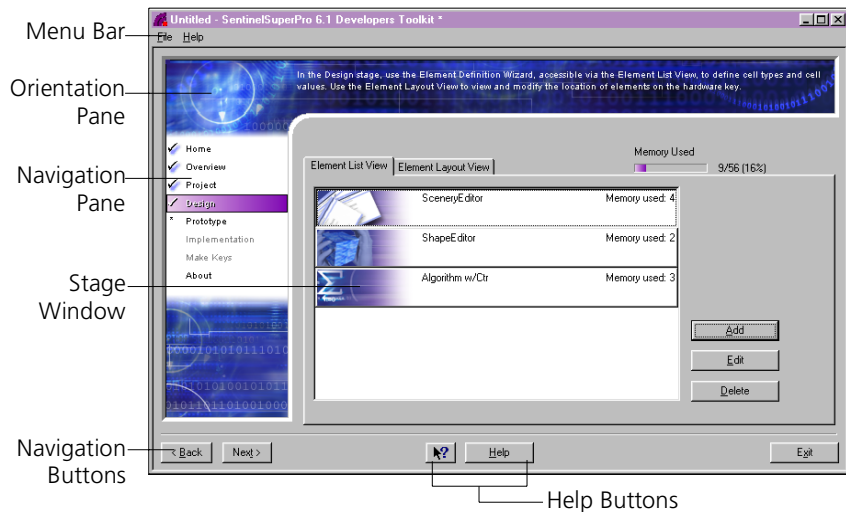
If you have already created projects with the Toolkit, be sure to read the warning message that appears thoroughly and take the appropriate action. See “About the Field Exchange DLLs” on page 114 for more information.

The Toolkit window appears.

# Navigating in the SentinelSuperPro Toolkit

When you open the SSP Toolkit, the toolkit window appears with the Home stage open. This window contains the following components:

- Stage window (stages and sections)
- Navigation pane and buttons
- Orientation pane
- Menu bar
- Help buttons



**Toolkit Window (with Design Stage Open)**

## Stages and Sections

The SSP Toolkit is made up of eight different *stages*. Stages appear in the *stage window*, where sections and sub-sections within the window help you navigate to the tasks necessary to implement your protection strategy. The stages are as follows:

- **Home** – The default stage that appears when the SSP Toolkit opens. No tasks are performed in this stage.
- **Overview** – Sections in this stage introduce you to SentinelSuperPro concepts. This stage also features the API Explorer, where you can test API function calls, view the key's cell layout, and send queries to the key to obtain return values.
- **Project** – This stage provides setup and configuration information. Create or open projects and enter your developer ID and passwords in this stage.
- **Design** – The Design stage has two sections: Element List View and Element Layout View. Use the Element Definition Wizard, accessible via Element List View, to define cell types and cell values. Element Layout View allows you to view and modify the location of algorithm, counter and data word cells on the hardware key.
- **Prototype** – In this stage, you program the cells in the hardware key with the values defined in the Design stage, generating pseudocode for use in adding API functions to your source code. *This stage is a required stage.*
- **Implementation** – When you implement your strategy, you add the appropriate protection to your application code, either by adding a shell to the application's executable file, or adding API functions to the source code based on the pseudocode generated during prototyping. This stage also allows you to define the actions that can be taken through field activation, and is used to create license codes for distribution to customers who have purchased upgrades in the field.
- **Make Keys** – Hardware keys programmed with your protection strategy, as defined in the Design stage, must be distributed with each copy of your software. The Make Keys stage allows you to program keys prior to distribution.
- **About** – For more information about the version of the SSP Toolkit you are using, or links to Rainbow Technologies information on the Web, go to this stage. No tasks are performed in this stage.



The following table defines the tasks required to implement your protection strategy and what stage they are located in:

**SentinelSuperPro Toolkit Tasks and Corresponding Stages**

Stage	Task
<b>Home</b>	No tasks are performed in this stage.
<b>Overview</b>	Tasks performed in this stage are for informational use only and are not required to implement a protection strategy.
<b>Project</b>	Create a new project or open existing project.
	Enter your developer ID and passwords.
	Save an existing project.
<b>Design</b>	Select integrated or automatic protection type.
	Select an activation type.
	Define time/date/execution controls (for demos only).
	Add counters, passwords, sublicense limits and/or data words.
<b>Prototype</b>	Program attached hardware key memory cells.
	Generate pseudocode and field exchange data.
<b>Implementation</b>	Shell applications.
	View pseudocode and add appropriate API functions to application source code.
	Define field activation actions and commands.
	Generate license codes based on locking codes received from customers in the field who have purchased upgrades.
<b>Make Keys</b>	Program product keys for distribution to customers and distributor keys for use by distributors who will activate your application.
<b>About</b>	No tasks are performed in this stage.

### ***Moving From Stage to Stage***

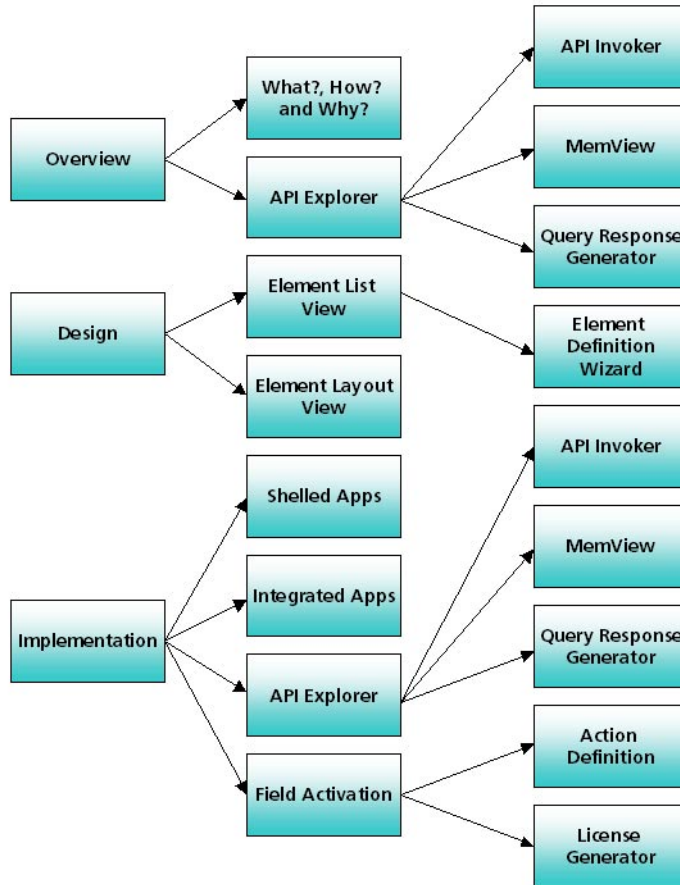
To move from stage to stage, click on the stage name in the *navigation pane*. Once you have visited a stage, a check mark appears to the left of the stage name. Your navigation history for the current project is saved in the project file.

You can also use the **Back** and **Next** navigation buttons, located beneath the navigation pane, to move sequentially through the stages.

Stages are arranged in the order you will typically use them; however, you do not need to visit the stages in this order, nor do you need to complete all the sections in each stage before moving to another stage. For example, you can go to the Project stage without first going to the Overview stage.

There is one exception to this rule. **The Prototype stage is a required stage** (identified by an asterisk [\*] to the left of the stage name). You cannot go to the Implementation or Make Keys stages until you have *completed* the Prototype stage.

The following diagram shows how you can move from stage to stage and section to section within the SSP Toolkit:



**Navigation Flow Through the SentinelSuperPro Toolkit**

## Menu Bar

SSP Toolkit commands are located in menus at the top of the toolkit window. When a command appears dimmed, it is unavailable for use with the selected stage or section. Menu options are described in the following table:

SentinelSuperPro Toolkit Menu Commands

Menu	Command	Description
<b>File</b>	New	Creates a new SuperPro project.
	Open	Opens an existing SuperPro project.
	Save	Saves the current project.
	Save As	Saves the current project under a new file name.
	Export DST File	Saves the current project as a .DST file. Provides limited project information to distributors.
	Exit	Exits SentinelSuperPro.
<b>Help</b>	Help Topics	Accesses the SentinelSuperPro online Help.
	Index	Accesses the online Help index.
	What's This?	Accesses context-sensitive help.
	About	Displays SentinelSuperPro version and copyright information.

## Getting Help

There are several ways to get help while using the SSP Toolkit. For general issues, look for answers in this guide and in the online Help system that is included with the SSP Toolkit.

You may also want to read through the text provided in the SSP Toolkit's Overview stage. The introductory information included there can help you gain a basic understanding of SentinelSuperPro concepts.

Additionally, as you move through the stages, pay attention to the text that appears in the *orientation pane* at the top of the SSP Toolkit window. This text provides a quick overview of the steps you'll take in each stage and how they apply to protection strategies. If you find yourself unsure of what to do in a particular stage, read the orientation pane text for help.


## Using Online Help

The SSP Toolkit ships with a complete online Help system. It includes a detailed table of contents and thorough index searching capabilities.

SSP Toolkit Help is very easy to use. The majority of the information found in this guide is also available through Help.

To access online Help, select **Help Topics** from the **Help** menu.

Most fields also have context-sensitive help, which is accessible by right-clicking on a field, check box or button to view Help information specific to that item.

Or, click the **What's This** button  at the bottom of the SSP Toolkit window to access the Help pointer, then click on the item you need help for.

For more information, please review the Using Help topic in online Help.

---

## Completing the Overview Stage

### Learning About SentinelSuperPro Concepts

The first three tabs in the Overview stage provide information about SentinelSuperPro concepts. The text in this section is designed primarily for those users who may not have access to the SentinelSuperPro documentation, yet need to get started protecting an application.

If you want a quick introduction to how SentinelSuperPro works, you may want to review this information. However, if you have read Chapters 1–4 in this manual, you can skip this information, as most of the concepts were explained in those chapters.

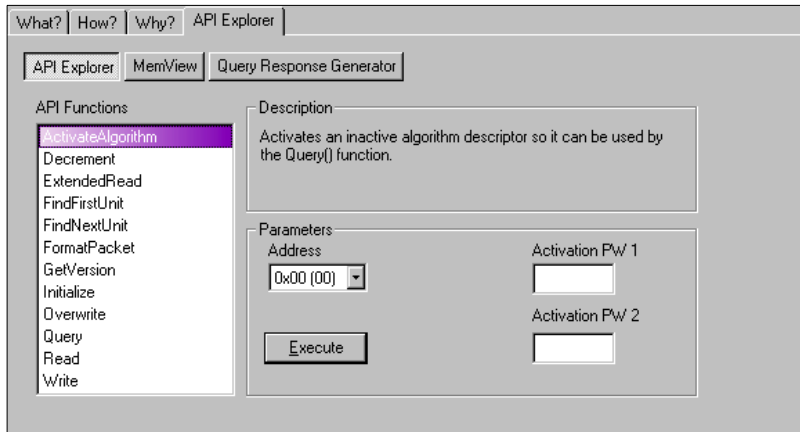
### Using the API Explorer

The API Explorer allows you to experiment with API function calls on various cells in the key before you add them to your source code. It is also a good way to familiarize yourself with the available functions and their uses prior to designing your strategy.

## Invoking API Functions

To invoke an API function on a selected cell:

1. Navigate to the **Overview** stage.
2. Click the **API Explorer** tab. The API Calls section appears.



### API Explorer Tab – API Calls Section

3. From the API function list, select a function. A description of the function appears to the right.
4. Under **Parameters**, select values for the available parameters.

Different functions have different parameters available; some functions do not have any parameters. The following is a list of possible parameters:

- **Address:** The cell you want to test the function on.
- **Activation Password:** The value used to activate an inactive algorithm.
- **Access:** The access code you want to write to the cell selected in the Address parameter.

- **Value:** The hexadecimal value to write to the selected cell. Use the Numeric Assistant to convert a decimal value to hexadecimal, or select a random value. See “Converting Decimal or Binary Values to Hexadecimal” on page 126 for more information.
- **Query Data:** The query string to send to the cell for encryption and a response value.

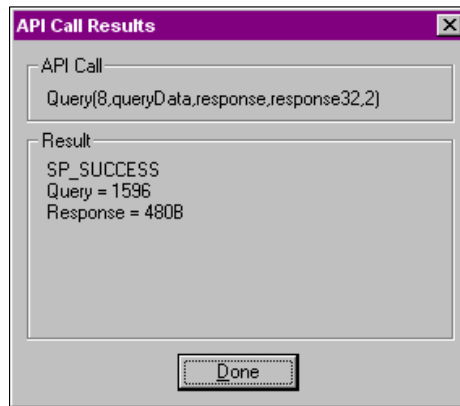
---

**Tip:** For more information about activation passwords, access codes, query data and response values, please see Chapter 3, “Using the Hardware Key,” on page 33.

---

5. Click **Execute**.

The API function is invoked, with the parameters you selected. The API Call Results message box appears:



**API Call Results Message Box**

Refer to “API Status Codes” on page 315 for a list of error codes that may appear in the Results list.

6. Click **Done** to close the message box.
7. Repeat steps 3 through 6 to test additional API functions.

---

**Note:** Remember, you need to call the `RNBOSproFormatPacket()` and `RNBOSproInitialize()` functions prior to calling any other function.

---

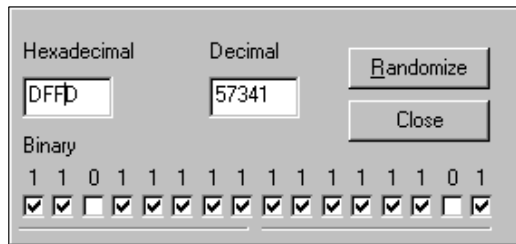
## Converting Decimal or Binary Values to Hexadecimal

When entering a value to be written to a cell, all values must be in *hexadecimal* format. The Numeric Assistant can be used to convert values from Base 2 (binary) or Base 10 (decimal) formats to the Base 16 (hex) format.

The Numeric Assistant can also randomly generate a value for you.

To access and use the Numeric Assistant dialog box:

1. In the **Value** field, click the down arrow. The Numeric Assistant dialog box appears.



**Numeric Assistant Dialog Box**

2. Do any of the following, as necessary to convert or generate your value:
  - To convert a decimal value to a hexadecimal value, enter the value in the **Decimal** field.  
  
The Numeric Assistant automatically calculates the corresponding hexadecimal and binary values.
  - To convert a binary value to a hexadecimal value, clear or select the check boxes located along the bottom of the dialog box. Each check box represents a binary digit—select a check box for the value **1**, clear a check box for the value **0**.



The Numeric Assistant automatically calculates the corresponding hexadecimal and decimal values.

- To generate a random value, click **Randomize**.

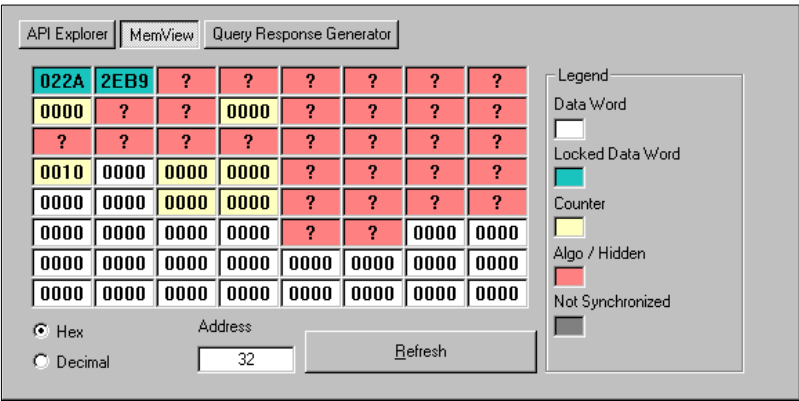
The Numeric Assistant generates a random value, and provides you with the hexadecimal, decimal and binary equivalents of that value. You can click Randomize as many times as you like.

3. Click **Close**.

The value from the Hexadecimal field is transferred to the Value field.

### Viewing Memory Cells

The MemView section of the API Explorer provides a graphical view of the address, access code and value for each cell on the attached key. You can also invoke API function calls on specific cells from this section.



### MemView – Programmed Key With Counters and Algorithms

To graphically view information about cells in an attached key:

1. Navigate to the **Overview** stage.
2. Click the **API Explorer** tab.

3. Click **MemView**. The MemView section appears with all cells shaded grey. This means the key has not yet been queried for the status of the programmed cells.
4. Click **Refresh**. The SSP Toolkit queries the key, returning the access code and value of each cell.

Cell access codes are identified by different colors, as shown in the legend on the right. Cell values are provided in hexadecimal format within each individual cell.

A question mark (?) represents a restricted cell, or a cell programmed with an algorithm value.

To view a selected cell's address, move the mouse pointer over that cell. The address appears in the Address field automatically.

---

**Tip:** To view cell values and addresses in decimal format, select **Decimal** in the lower left of the window.

---

To invoke an API function for a selected cell from the MemView section:

1. Right-click on the cell and select **API** and then the function from the shortcut menu that appears. The API Function dialog box appears.
2. Enter the appropriate parameters for the function. See page 124.
3. Click **Execute**.

The API function is invoked on the selected cell, and the return value appears in the API Call Results message box. See “Invoking API Functions” on page 124 for more information.

## Querying Algorithms

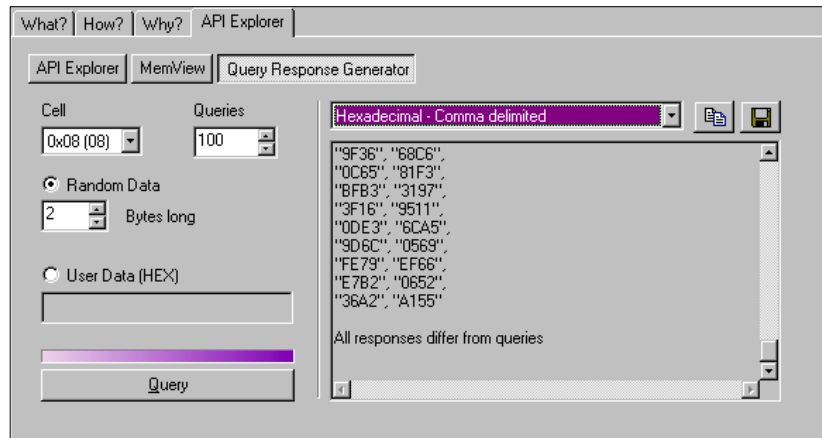
The Query Response Generator allows you to experiment with and learn about the different types of algorithms and their states.

When your application sends a query to an algorithm in your key, it compares the encrypted response to the response it expects. To determine the expected responses, you must query the algorithm during your development phase.

To query an algorithm word:

1. Navigate to the **Overview** stage.
2. Click the **API Explorer** tab.
3. Click **Query Response Generator**.

The Query Response Generator screen appears.



### Query Response Generator

4. From the **Cell** drop-down list, select an algorithm word to query.

Use the MemView section to determine which cells on your key are algorithm cells. See “Viewing Memory Cells” on page 127.

5. In the **Queries** box, enter or select the number of queries you want to make.
6. Do one of the following:
  - To use random data generated by the SSP Toolkit as a query string, select **Random Data**, then select how long, in bytes, you want the query string to be. The query string must be between 1 and 20 bytes.
  - To use your own data as a query string, select **User Data**, then enter the data in the corresponding field. The data must be in hexadecimal format with an even number of digits, up to 10.
7. Click **Query**.

The SSP Toolkit queries the algorithm cell you selected in step 4 with the query string selected in step 6.

The result values appear—in hexadecimal format—in the query/response text box. Select an option from the drop-down list to view the return values in other formats. The value in the Response column is the encrypted string your application should expect to receive when it sends the same query.

A summary is included at the end of the list to help you determine if the algorithm is active or inactive. If the responses are the same as the queries, the algorithm is inactive. If the responses are different than the queries, the algorithm is active.

8. To query a new cell, repeat steps 4 through 7.

---

# Creating a Project

## What Is a Project?

A project is stored in a SentinelSuperPro Toolkit file. The project contains all the data used to create your protection strategy—elements, passwords, your developer ID, algorithm values, counters, data words, field activation commands, etc.

Your project is the template that will be used to program the keys protecting your application.

Projects are protected with strong encryption, so your passwords and developer ID cannot be obtained from a project without opening it in the SSP Toolkit.

Additionally, you can prevent unauthorized users from being able to open your projects (even if they have the SSP Toolkit) by locking them. See “Adding Password Protection to Your Project” on page 135 for more information.

Because your project contains sensitive information about your protection strategy, we recommend making it accessible to developers only—do not give your project file to manufacturing or distribution personnel *who also have access to the SSP Toolkit*.

---

**Note:** *Manufacturing personnel **do** need the project file to use the Make Keys Utility, but they will not be able to make changes to your project with these utilities. Distributors should be given a .DST file for use with the License Generator Utility. See “Creating a Project File for Distributors” on page 138 for more information.*

---

Projects can be saved and re-opened for editing as needed.

## Creating a New Project

When you open the SSP Toolkit, a new project, *untitled.spp*, is created by default. We recommend saving this project with a more meaningful name before starting to design your protection strategy. See “Saving Your Project” on page 135 for instructions.

You may also want to create a new project after you have opened an existing project. To do so:

1. Navigate to the **Project** stage.
2. Click **New**.
3. If another project is open, and you haven’t saved it, you are asked if you want to save the current project. Click **Yes** to save your changes, or **No** to discard them.

A new untitled project opens.

## Importing a .DAT File

If you have used previous versions of SentinelSuperPro, you may have already created protection strategies. You can edit existing strategies in the SSP 6.1 Toolkit by importing the .DAT file created by previous versions.

The <profile\_name>.DAT file contains the data used to program your keys, and was created using the SentinelWizard in previous versions of Sentinel-SuperPro.

---

**Note:** *.DAT files were created by SentinelSuperPro versions 5.1 and earlier. Projects created using SentinelSuperPro 6.0 were saved with an .SPP extension, and may be opened using the procedure on page 134. Files saved in NetSentinel cannot be opened in SentinelSuperPro 6.1.*

---

To import a .DAT file:

1. Navigate to the **Project** stage in a new project.
2. Click **Import DAT File**. The Open dialog box appears.

3. Browse to locate the .DAT file you want to import, then click **Open**.

The .DAT file, with the elements you defined in the previous version, is imported into the current project. The elements are added to the current project. If there are cell conflicts during the import process, a warning message appears and the element that caused the conflict is not imported.

You can merge two .DAT files into one project file by importing both .DAT files into the same project file.

## Changing Your Developer ID or Passwords

When creating a project, you may need to change the developer ID and/or passwords (for example, if you are using a different SentinelSuperPro key).

See “Entering Your Passwords” on page 112 for more information about your developer ID, the overwrite passwords and the write password.

To change your developer ID and/or passwords:

1. Navigate to the **Project** stage.
2. Click **Configure**. The Developer Configuration dialog box appears.
3. In the **Developer ID** field, enter your developer ID.

---

**Tip:** *If you want to be able to see the actual password and developer ID characters, select the **Show Passwords** check box. Password characters are hidden (displayed as asterisks) by default.*

---

4. In the **Overwrite Password 1** field, enter the first overwrite password for your key.
5. In the **Overwrite Password 2** field, enter the second overwrite password for your key.
6. In the **Write** field, enter the write password for your key.
7. Click **OK**. A warning message appears.

8. Read the warning message thoroughly, and take the appropriate action. See “About the Field Exchange DLLs” on page 114 for more information.

You are returned to the Project stage.

---

## Opening an Existing Project

Once you have saved a project, you can re-open it later to edit it, program keys, or generate license codes for customer upgrades.

To open an existing project:

1. Navigate to the **Project** stage.
2. Click **Open**.

The Open dialog box appears.

3. Browse to locate the SentinelSuperPro project you want to open, then click **Open**.

All SentinelSuperPro projects have an extension of .SPP.

4. If you haven't saved the currently open project, you are asked if you want to save the current project. Click **Yes** to save your changes, or **No** to discard them.

If the project was locked, the Password dialog box appears. Enter the unlock password. The project opens in the SSP Toolkit.

See “Adding Password Protection to Your Project” on page 135 for more information about locked projects.



---

## Saving Your Project

We recommend saving your project often, particularly when you first open or before you close the SSP Toolkit.

- To save your project with the existing name, from the **File** menu, select **Save**.

---

**Tip:** You can verify the name of the project you are currently viewing by looking at the SentinelSuperPro window title bar, where the name of the project is displayed.

---

- To save the project under another file name:
  1. From the **File** menu, select **Save As**. The Save As dialog box appears.
  2. Enter the new project name in the **File Name** field, then click **Save**. The project is saved under the new file name.

---

## Adding Password Protection to Your Project

To protect against unauthorized access to your project, we recommend you *lock* your project. Locking adds password protection to your project, so that you must enter a password to open it in the SSP Toolkit.

Password-protecting your project is particularly important, as whoever has access to your project file also has access to your developer ID, write passwords, commands and actions.

When you open a locked project, you have three tries to enter the correct password. If the correct password is not entered after three attempts, you are locked out of the project. Open the project again to continue trying to enter the correct password.

---

**Warning!** *If you forget your password, you will need to recreate your project. There is no “back-door” that Rainbow Technologies Technical Support can use to give you access to your project. Thus, it is **VERY** important that you remember the password you use to lock your project.*

---

## Locking a Project

1. In the SSP Toolkit, open the project you want to lock.
2. Navigate to the **Project** stage.
3. Click **Lock**. The Password dialog box appears.

A screenshot of a 'Password' dialog box. The dialog box has a purple title bar with the text 'Password' and a close button (X). The main area is light gray and contains three text input fields labeled 'Old Password', 'New Password', and 'Confirm Password'. At the bottom, there are two buttons: 'OK' and 'Cancel'.

### Password Dialog Box

4. In the **New Password** field, enter the password you want to use to lock the project. Passwords are case-sensitive and are limited to 12 characters.
5. In the **Confirm Password** field, enter the same password again for confirmation.
6. Click **OK**. The project is locked. The next time you open it, you will be required to enter the password you selected in step 4.

## Changing the Password for a Locked Project

You can change the password for a locked project at any time, as long as you know the existing password. To change the password:

1. Open the project whose password you want to change.
2. Navigate to the **Project** stage.
3. Click **Lock**. The Password dialog box appears.
4. In the **Old Password** field, enter the existing password.
5. In the **New Password** field, enter the new password.
6. In the **Confirm Password** field, enter the new password again for confirmation.
7. Click **OK**. The password for the locked project is changed. The next time you open the project, you will need to enter the new password.

## Unlocking a Project

To unlock a project and remove the password protection:

1. Open the project you want to unlock.
2. Navigate to the **Project** stage.
3. Click **Unlock**. The Password dialog box appears.
4. Enter the password for the locked project, then click **OK**.

Password protection is removed from the project and the Unlock button becomes unavailable, indicating the project is no longer locked.

## Creating a Project File for Distributors

To avoid giving your distributors access to your passwords—which would also give them the ability to change field activation commands or other elements in your protection strategy—we recommend creating a project file specifically for your distributors.

This special file, a .DST file, allows your distributors to activate and update product keys, but prevents them from making changes to your strategy. The .DST file also prevents distributors from increasing the number of licenses available on their distributor keys by incrementing the distributor counter.

Your distributor will need to open the .DST file in the License Generator Utility in order to generate license codes and activate or update product keys. For more information about how the distributor updates keys using this file, see “How Distributors Activate an Application” on page 247.

To create a .DST file:

1. In the SSP Toolkit, open the project you want to create a distributor’s file from.
2. While in any stage, from the **File** menu, select **Export .DST File**. The Save As dialog box appears.
3. Enter the distributor file name in the **File Name** field, then click **Save**.

The distributor file is saved under the file name you entered, with a .DST extension. It is now ready to be sent to your distributors.

---

## Closing the SentinelSuperPro Toolkit

To close the SSP Toolkit:

1. From the **File** menu, select **Exit**.
2. If you have made changes to your project, but haven't saved them, you are prompted to do so. Click **Yes** to save your changes or **No** to discard them.



# Chapter 7

---

## Protecting Your Application

When you start designing your protection strategy, the first decision you need to make is whether you want to protect your application using one of the predefined protection types—integrated or automatic—or if you want to add your own custom elements.

Custom elements are individual algorithms, counters, sublicense limits or data words. You define not only the values for these elements, but also their location on the key. For more information about using custom elements, see Chapter 8, “Working With Design Elements,” on page 169.

This chapter explains how to protect your application using one of the predefined application protection types. We recommend selecting and implementing one of these types for every application you are protecting. If you want to also use custom elements, you can do so in addition to the standard application protection.

This chapter covers the following topics:

- What is application protection?
- Selecting a protection type
- Using integrated protection
- Using automatic protection

---

## What Is Application Protection?

Application protection is an algorithm with an associated activation type as determined by the options you choose to include in your strategy. Application protection can be either *integrated* or *automatic*. The protection type determines when and where software locks are implemented.

When you choose *integrated* protection, you add software locks—API functions—directly into your application’s source code. You control the amount and location of the locks.

When you use *automatic* protection, SentinelSuperPro wraps a protective layer, called a *shell*, around your application’s executable file. This layer is encrypted, making it more difficult for a hacker to gain access to your application’s code.

Whatever level of protection you decide to implement, the overall goal is to have your application periodically check that the key is present. As long as your application takes appropriate measures if the key is not attached, only legitimate purchasers will be able to use your application.

Both stand-alone and network applications can use either protection type. However, an application using automatic protection will always use the default dual access mode. See “Setting the Access Mode” on page 100 for more information about access modes.

---

**Note:** *If you apply both automatic and integrated protection to your application, when the automatic portion of the application executes it will run in dual mode. When the integrated portion of the application executes, it will run in the access mode you set in code, or dual mode if no access mode was specified. See “Setting the Access Mode” on page 100 for more information.*

---

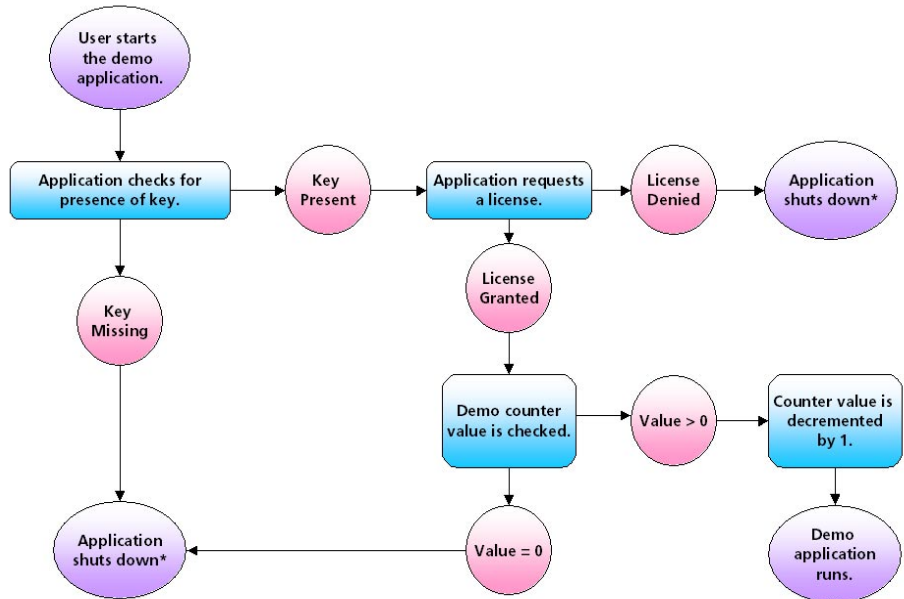
For more information about application protection types, and examples of when you should choose one over the other, please see “Protection Types” on page 57.



## Demo Applications

With both types of application protection, you can designate your application as being a *demonstration (demo)* or trial version through the use of a counter that controls the number of times the application can run before it expires.

For example, set the demo counter value to 5. Each time the application is run, the counter is decremented by one. The sixth time the user tries to run the application, she will be unable to, because the execution counter has expired, which deactivates the algorithm, and the algorithm returns an invalid response.



*\*The action to take if a key is missing or the license is denied is up to the developer.*

## How a Demo Application Runs

If you choose to use automatic protection, SentinelSuperPro provides increased control over demo applications. In addition to the execution counter, you also have the ability to control the length of time or number of days the application can be run. You can also define a static expiration date for the demo application.

When an application (using either protection type) is designated as a demo, its activation type must be *static* or *trusted*, so that it can be re-activated with a password when it expires and you have verified purchase of the full version of the application.

For more information about designating demo applications, see “Controlling Demo Applications” on page 90.

---

## Selecting a Protection Type

The first step in applying application protection is to select whether you want to use automatic or integrated protection. The Element Definition Wizard, accessible via the Element List View tab, will walk you through the process of adding protection to your application.

To add protection to your application:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click **Add** to start the Element Definition Wizard. You are asked to select the type of element you want to add.
4. Select **Application Protection**, then click **Next**.
5. Select **Integrated** or **Automatic**, then click **Next**.
6. In the **Name** field, enter a name for this element.

We recommend using the name of the application you are protecting. There is a 16-character limit for element names.

For example, if you are using automatic protection to protect the SceneryEditor application, you might name the element *Scenery (Auto)*.

This way, when viewing the element list, you'll quickly be able to recognize what applications you have protected. The icons in the element list show you what kind of application protection is used for each application.

---

**Tip:** *This name will be used throughout the SSP Toolkit to identify this element, so be sure it adequately describes the element.*

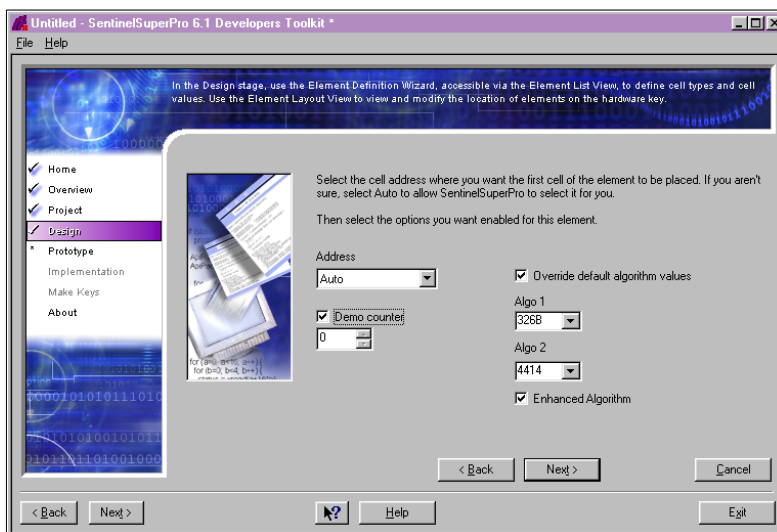
---

7. In the **Comments** field, enter any additional information about the element you want to save. This field is optional.
8. Click **Next**.
9. Do one of the following:
  - If you selected **Integrated**, go to the next section to continue.
  - If you selected **Automatic**, go to “Using Automatic Protection” on page 152 to continue.

---

## Using Integrated Protection

Complete the steps in the following sections to finish defining your application protection.



### Defining Integrated Protection Options

## Selecting the First Cell

1. From the **Address** drop-down list, select the address of the cell you want the first word of the element to be placed in.

If the location is unimportant to you, select **Auto** to allow the SSP Toolkit to select a location for you.

---

**Note:** *Throughout the SSP Toolkit, only valid and available cell addresses are provided in Address drop-down lists, preventing you from selecting an inappropriate address.*

---

2. Go to the next section.

## Adding a Demo Counter

This section is optional. If you *don't* want to make this application a demo application, skip this section and go to “Overriding the Default Algorithm Values” on page 148.

If you want to make this application a demo application that can be used for a limited number of times, as explained in “Demo Applications” on page 143, do the following:

1. Select the **Demo Counter** check box.
2. In the counter value box, enter a number representing the number of times you want to allow the demo application to be executed.

For example, if you enter **5**, the application can be run five times. The sixth time the user tries to run the application, they will be unable to.

3. Click **Next**.
4. Go to the next section.

## Overriding the Default Algorithm Values

SentinelSuperPro generates random algorithm values. We recommend that you accept the default algorithm values.

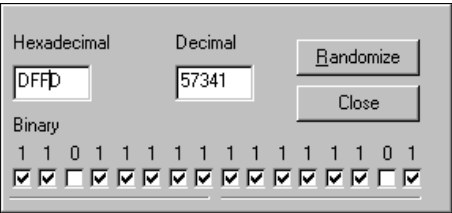
To accept the default algorithm values, skip this section and go to “Selecting the Activation Type” on page 149.

If you want to define your own algorithm values, review the information in “Algorithm Values” on page 45, then do the following:

1. Select the **Override default algorithm values** check box.

The algorithm value fields appear.

2. In the **Algo 1** field, click the arrow button to access the Numeric Assistant dialog box.



**Numeric Assistant Dialog Box**

3. Do any of the following, as necessary to generate your hexadecimal value:
  - To generate a random value, click **Randomize**.  
The Numeric Assistant generates a random value, and provides you with the hexadecimal, decimal and binary equivalents of that value. You can click Randomize as many times as you like.
  - To convert a decimal value to a hexadecimal value, enter the value in the **Decimal** field.  
The Numeric Assistant automatically calculates the corresponding hexadecimal and binary values.

4. Click **Close**.

The value from the Hexadecimal field is transferred to the Algo 1 field.

5. Repeat steps 2 – 4 for the **Algo 2** field.

---

**Note:** *If you already know the value you want to use for the algorithm words, you can enter it directly in the **Algo 1** and **Algo 2** fields. Algo 1 is the first algorithm word and Algo 2 is the second algorithm word. When you select an activation type, the Algo 2 value will be automatically changed, as necessary, to make the algorithm active or inactive.*

---

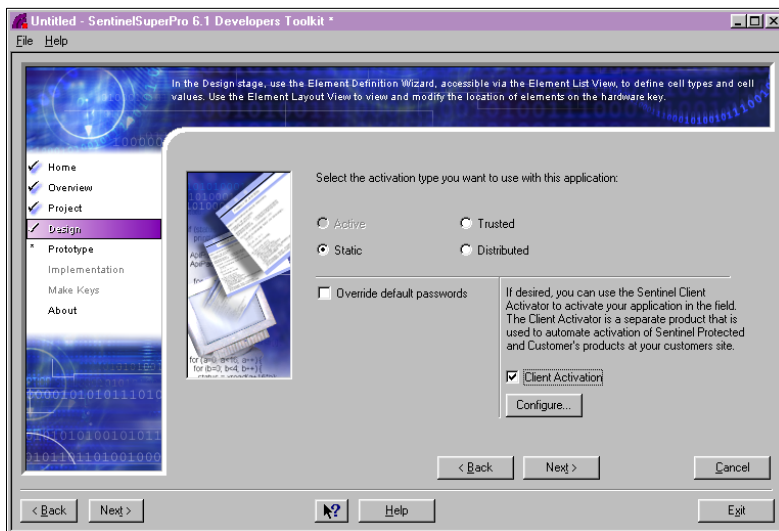
6. If you want this algorithm to use the enhanced algorithm engine, select the **Enhanced Algorithm** check box.

For maximum security, use of the enhanced algorithm engine is recommended for all algorithms.

7. Go to the next section.

## Selecting the Activation Type

Next, you need to choose an activation type to use. If you added a demo counter, the default activation type is *Active*. If you did not add a demo counter, the default activation type is *Static*.



### Selecting Activation Type Options

To select an activation type:

1. Select the activation type you want to use for this application: **Active**, **Static**, **Trusted** or **Distributed**.

---

**Tip:** For more information about activation types, including when to use each type, please see “Activation Types” on page 60.

---

2. Do one of the following:
  - If you chose **Active**, go to step 6.
  - If you chose **Static** and want to override the default activation passwords, select the **Override default passwords** check box. The activation password fields appear. Go to step 3.
  - If you chose **Trusted** or **Distributed**, or you chose **Static** but don't want to override the default activation passwords, go to step 5.



---

**Note:** *If you selected Trusted or Distributed, you cannot override the default activation passwords. Unique activation passwords are generated based on the developer ID, serial number and product information. See the table “SentinelSuperPro Activation Types” on page 60 for more information about the Trusted and Distributed activation types.*

---

3. In the **Password 1** field, click the arrow button to access the Numeric Assistant dialog box, and enter an activation password for the first word of the algorithm.

See page 148 for instructions on using the Numeric Assistant dialog box.

4. Repeat step 3 for the **Password 2** field to enter a password for the second word of the algorithm.
5. Do one of the following:

- If you want to use the Client Activator to develop product-specific activation information for the application, select the **Use Client Activator** check box, then click **Configure** to launch the Activation Wizard.

When you have completed defining your application’s activation information, save your project and close the Activation Wizard to return to the SSP Toolkit.

- If you don’t want to use the Client Activator, go to the next step.

---

**Tip:** *For more information about the Client Activator, see “Using the Client Activator” on page 249. You may also want to refer to the Client Activator documentation, included in the Client Activator package.*

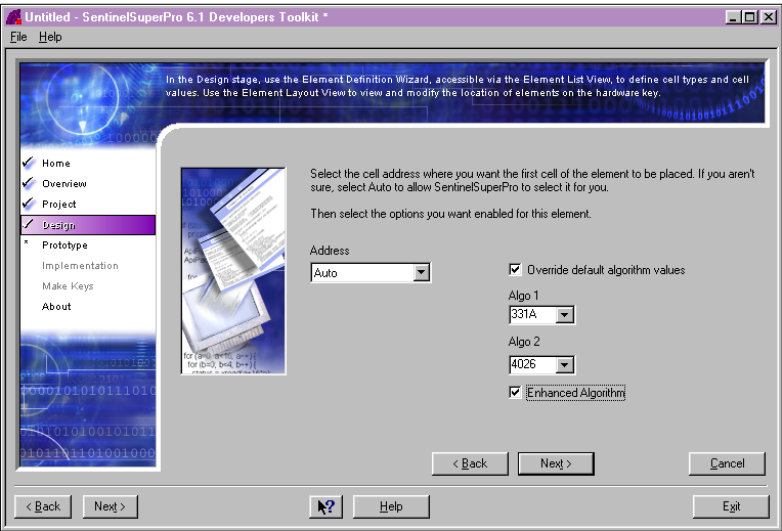
---

6. Click **Next**.

Element definition is complete and you are returned to the Element List View tab, where your application now appears in the list.

# Using Automatic Protection

Complete the steps in the following sections to finish defining your application protection.



## Defining Automatic Protection Options

### Selecting the First Cell

1. From the **Address** drop-down list, select the address of the cell you want the first word of the element to be placed in.

If the location is unimportant to you, select **Auto** to allow the SSP Toolkit to select an appropriate location for you.

---

**Note:** Be sure to review “Valid Algorithm Addresses” on page 47 for more information about where elements can be placed on the key.

---

2. Go to the next section.

## Overriding the Default Algorithm Values

SentinelSuperPro generates random algorithm values. We recommend that you accept the default algorithm values.

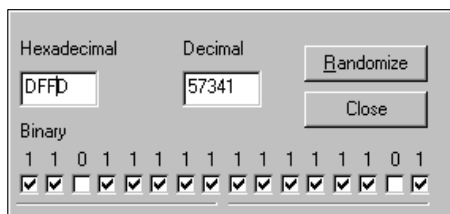
- If you want to accept the default values, skip this section—go to “Selecting the Input and Output Files” on page 154.
- If you want to define your own algorithm values, review the information in “Algorithm Values” on page 45, then return to this procedure.

To select your own algorithm values:

1. Select the **Override default algorithm values** check box.

The algorithm value fields appear.

2. In the **Algo 1** field, click the arrow button to access the Numeric Assistant dialog box.



The Numeric Assistant dialog box is a small window with a light gray background. It contains three input fields: 'Hexadecimal' with the value 'DFFD', 'Decimal' with the value '57341', and 'Binary' with the value '1 1 0 1 1 1 1 1 1 1 1 1 0 1'. Below the 'Binary' field is a row of 14 checkboxes, each corresponding to a bit in the binary string. The first 13 checkboxes are checked, and the last one is unchecked. To the right of the input fields are two buttons: 'Randomize' and 'Close'.

### Numeric Assistant Dialog Box

3. Do any of the following, as necessary to generate your value:
  - To generate a random value, click **Randomize**.

The Numeric Assistant generates a random value, and provides you with the hexadecimal, decimal and binary equivalents of that value. You can click Randomize as many times as you like.

- To convert a decimal value to a hexadecimal value, enter the value in the **Decimal** field.

The Numeric Assistant automatically calculates the corresponding hexadecimal and binary values.

4. Click **Close**.

The value from the Hexadecimal field is transferred to the Algo 1 field.

5. Repeat steps 2 – 4 for the **Algo 2** field.

---

**Note:** *If you already know the value you want to use for the algorithm words, you can enter it directly in the **Algo 1** and **Algo 2** fields. Algo 1 is the first algorithm word and Algo 2 is the second algorithm word. When you select an activation type, the Algo 2 value will be automatically changed, as necessary, to make the algorithm active or inactive.*

---

6. If you want this algorithm to use the enhanced algorithm engine, select the **Enhanced Algorithm** check box.

For maximum security, use of the enhanced algorithm engine is recommended for all algorithms.

7. Go to the next section.

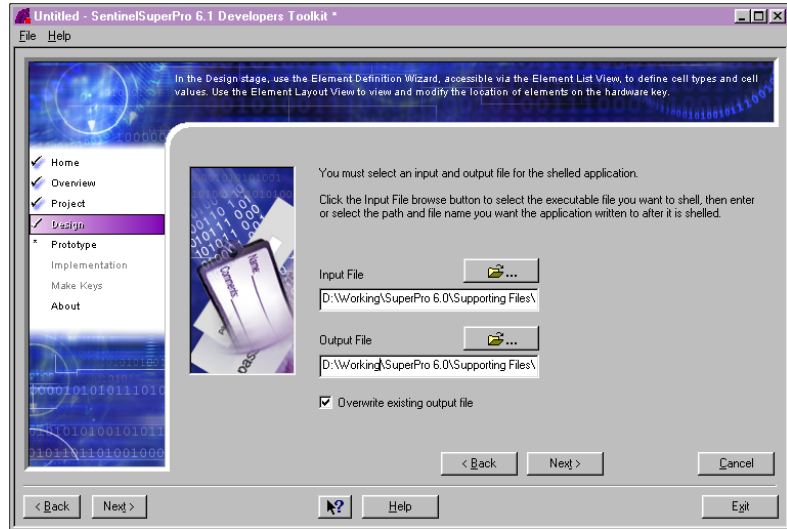
### Selecting the Input and Output Files

Next, you need to select the executable file (.EXE or .DLL) you want to apply the shell to. This file is the *input file*. You also need to select the path and file name you want the *protected* application's executable file written to. This is the *output file*—the file you will ship to your customers. You must select both an input file and an output file.

---


**Note:** *If you select files to be encrypted at shell time, and your application will be run on Windows 95 or 98, you must install the Sentinel data protection driver on your user's system. See "Installing the Sentinel Data Protection Driver" on page 236 for more information.*

---



### Selecting the Input and Output Files

To select the input and output files:


1. Click the browse button  located above the **Input File** field. The Open dialog box appears.
2. Browse to locate and then select the executable file you want to protect, then click **Open**.

The executable file's path appears in the Input File field.

---

**Note:** There is a 127-character limit for the path of the input or output files. If the file you want to shell is in a path that exceeds this limit, an error message will appear when you click Next. Move the file to a path with shorter directory names, then update the location and click Next again to continue.

---

3. Click the browse button  located above the **Output File** field. The Open dialog box appears.
4. Browse to locate the directory path you want the protected executable file placed in after the shell is added. Be sure to enter a file name for the shelled executable file at the end of the path.
5. If you want the shelled file to overwrite an existing file with the same name, select the **Overwrite existing output file** check box.

---

---

**Warning!** *If you specify the shelled application's executable file to have the same name as the non-shelled file, **and** you select the overwrite option, the non-shelled file will be overwritten with the shelled file.*

*We recommend changing the name of the output file to something different than the original file to preserve an original, unshelled copy of your application's executable file. You may also want to make a backup copy of your unshelled executable file.*

---

---

6. Go to the next section.

## Selecting Automatic Protection Demo Options

Using automatic protection gives you more control over demo applications than integrated protection. You have three options for limiting the execution of an application:

- You can limit the number of **executions** allowed.
- You can specify the last **date** the application can be run.
- You can limit the **time** allowed to use the demo application.

If you select two or three limits, the application will expire as soon as *any* limit is reached.

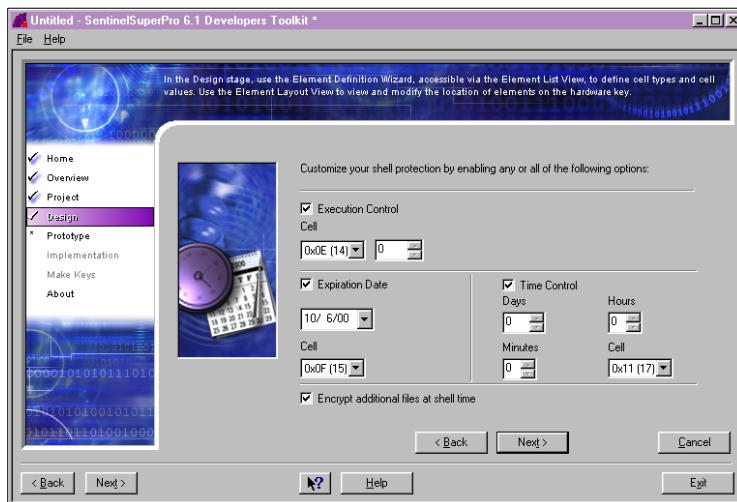
You must also select a cell in which to store the desired limit—SentinelSuperPro programs the limits you set into the cells you select. For execution

and time control, these cells are programmed as counters. For date control, the cell is programmed as a locked data word.

If your application uses execution or time control, it reads the value of the selected counter when it is run. Depending on the value, one of the following occurs:

- **Counter Value = 0:** The application is not allowed to start, and an error message appears. If time control is being used, the application is allowed to finish the current session.
- **Counter Value = 1–65534:** The counter is decremented every time the application is launched or time elapses, depending on the option selected.
- **Counter Value = 65535 (0xffff = -1):** The application runs without decrementing the counter. This allows the same executable to function as both a demo and an unlimited usage product.

This section also allows you to designate any application—not just demo applications—to check for the presence of the hardware key on a regular, predetermined basis.



## Selecting Demo Options

Use the following table to decide which options you want to use. You can select any, all or none of these options.

Automatic Protection Options

Option	Description	To Enable:
<b>Execution Control</b>	Allows you to limit the number of times the application can be run. Uses a counter cell that is decremented by one each time the application is run.	<ol style="list-style-type: none"><li>1. Select the <b>Execution Control</b> check box.</li><li>2. In the <b>Cell</b> field, select the cell you want the counter placed in. Select <b>Auto</b> if you want SentinelSuperPro to select a cell for you.</li><li>3. In the <b>Value</b> field, enter a number representing how many times you want to allow the application to be run.</li></ol>
<b>Expiration Date</b>	Allows you to control when the application can be run, by specifying a static expiration date. When the date is reached, the application will no longer run. Uses a data word cell that is queried each time the application is run. A second data word stores the date the application was last executed, to protect against date tampering.	<ol style="list-style-type: none"><li>1. Select the <b>Expiration Date</b> check box.</li><li>2. In the date field, click the arrow to access a calendar, then select the expiration date by clicking on it.</li><li>3. In the <b>Cell</b> field, select the cell you want the data placed in. Select <b>Auto</b> if you want SentinelSuperPro to select a cell for you.</li></ol>
<b>Time Control</b>	Allows you to control how long the application can be run for, in days, hours and/or minutes. The timer begins the first time the application is executed. Uses a counter cell.  For example, if you set the time control to 10 minutes, the counter cell value is 2 (two increments of 5 minutes each). When the user runs the application for the first time, the counter is decremented right away by one (5 minutes).	<ol style="list-style-type: none"><li>1. Select the <b>Time Control</b> check box.</li><li>2. In the <b>Days, Hours, and/or Minutes</b> fields, enter numbers representing how long you want the application to be active.</li><li>3. In the <b>Cell</b> field, select the cell you want the timing data placed in. Select <b>Auto</b> if you want SentinelSuperPro to select the cell for you.</li></ol>



**Automatic Protection Options (Continued)**

Option	Description	To Enable:
<b>Time Control (cont'd)</b>	<p>Five minutes later, the counter is decremented again to 0. At this point, the application will continue to run for an unlimited amount of time, until the user closes it. However, once the user closes the application, she won't be able to run it again because the counter = 0.</p> <p>Thus, it is possible for the application to run for a shorter or longer amount of time than that you identify in the time control.</p> <p>You can select a maximum of 226 days, 24 hours and 60 minutes.</p>	
<b>Encrypt additional files at shell time</b>	Allows you to encrypt additional files, such as data files, when you apply the shell to the application executable.	<ul style="list-style-type: none"> <li>• Select the <b>Encrypt additional files at shell time</b> check box.</li> </ul>

---

**Note:** In SentinelSuperPro 6.0, you were able to control how often the application checked for the presence of the key (the “background check” option). This option is now automatically set for all applications using automatic protection. The application will check for the presence of the key every minute; if the key does not respond, the application will shut down.

---

1. After you have finished selecting the options you want to use, click **Next** to continue.
2. Do one of the following:
  - If you chose to encrypt additional files at shell time, go to the next section.
  - If you chose not to encrypt additional files, go to “Selecting the Activation Type” on page 163.

## Selecting Additional Files for Encryption

The SSP Toolkit can also encrypt external files—such as overlays, data files, .DLLs or other file types—other than the application executable at shell time.

Your protected application will automatically and transparently decrypt at run time, as needed, the files you specified for encryption. When not in use, these files are re-encrypted. If your application creates one of these files, it will be encrypted.

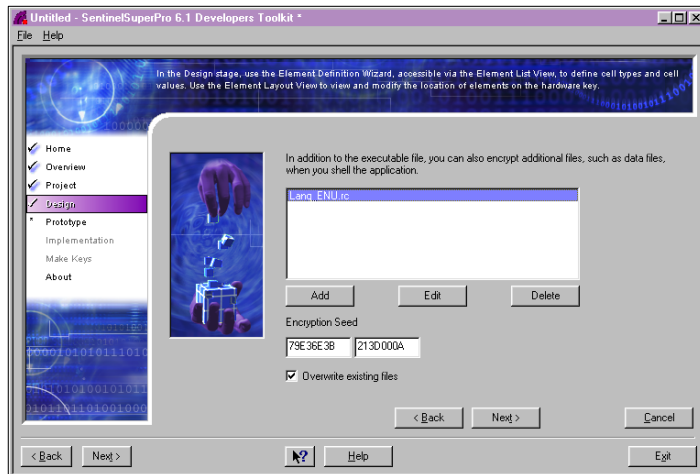
---

**Note:** *If you are shelling an application and you try to encrypt a .DLL which is a dependency of the application you are shelling, the .DLL will not be decrypted at runtime because the operating system tries to load the .DLL before the shell has a chance to decrypt it.*

---

The actual file encryption takes place when you add the shell to the application in the **Implementation** stage.

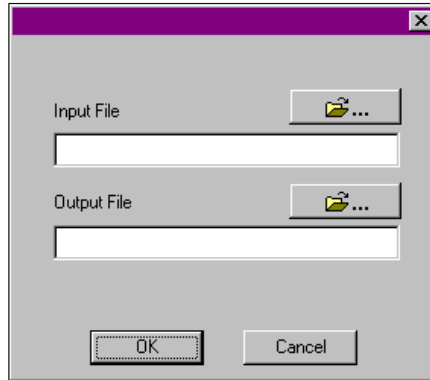
You can encrypt a maximum of 50 files at a time. To encrypt more than 50 files, add a second automatic protection element. In the second element, be sure to use the same encryption seed to encrypt the additional files with the same encryption.





Selecting Files for Encryption

To select files for encryption:

1. Click **Add**. The Add Files dialog box appears.



**Add Files Dialog Box**

2. Click the browse button  located above the **Input File** field. The Open dialog box appears.
3. Browse to locate and then select the file you want to encrypt, then click **Open**. The file's path appears in the Input File field.
4. Click the browse button  located above the **Output File** field. The Open dialog box appears.
5. Browse to locate the directory path you want the encrypted file placed in, then click **Open**. Be sure to enter a file name for the encrypted file at the end of the path.

---

---

**Warning!** We recommend changing the name of the output file to something different than the original file to preserve an original, unencrypted copy of the file. You may also want to create backup copies of the original files.

---

---

6. Click **OK**.

7. Repeat steps 1 – 6 to select additional files for encryption.

---

**Note:** *If you add a file and then decide you don't want to encrypt that file, select the file and then click **Delete**. You can also change the location path for an input and/or output file by selecting the file and then clicking **Edit**.*

---

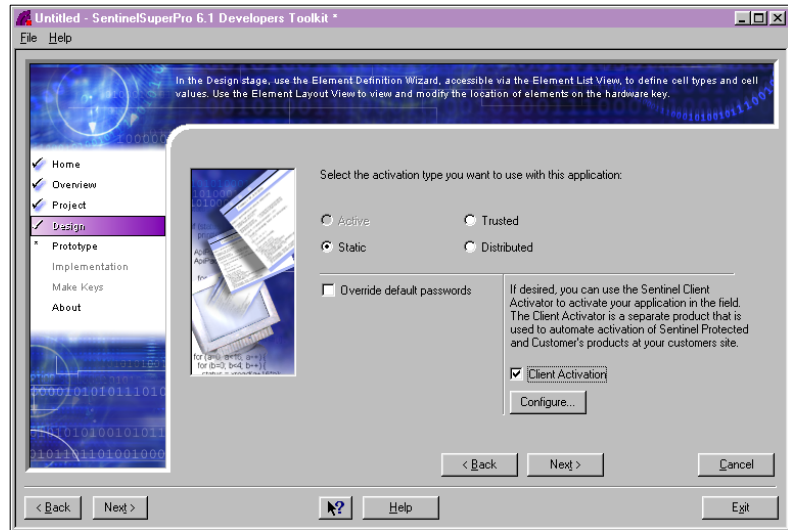
8. In the **Encryption Seed** fields, enter the seed (password) to be used for encryption. The seed is any two strings of eight hex characters (forming a 64-bit seed).

A random encryption seed is provided; you can choose to use this seed if you like. If encrypted data files are shared by multiple applications, all the applications must use the same encryption seed. See page 82 for more information about encryption seeds.

9. If you want encrypted files to overwrite existing files with the same name, select the **Overwrite existing files** check box.
10. Click **Next**, then go to the next section to continue.

## Selecting the Activation Type

Next, you need to choose an activation type to use. A suggested activation type, depending on the options you chose, is selected by default.



### Selecting Activation Type Options

---

**Note:** For more information about activation types, including when to use each type, please see “Activation Types” on page 60.

---

To select an activation type:

1. Select the activation type you want to use for this application: **Active**, **Static**, **Trusted** or **Distributed**.
2. Do one of the following:
  - If you chose **Active**, go to step 6.
  - If you chose **Static** and want to override the default (random) activation passwords, select the **Override default passwords** check box. The activation password fields appear. Go to step 3.

- If you chose **Trusted** or **Distributed**, or you chose **Static** but don't want to override the default activation passwords, go to step 5.

---

**Note:** *If you selected Trusted or Distributed, you cannot override the default activation passwords. Unique activation passwords are generated based on the developer ID, serial number and product ID. See the table “SentinelSuperPro Activation Types” on page 60 for more information about the Trusted and Distributed activation types.*

---

3. In the **Password 1** field, click the arrow button to access the Numeric Assistant dialog box, and enter an activation password for the first word of the algorithm.

See page 148 for instructions on using the Numeric Assistant dialog box.

4. Repeat step 3 for the **Password 2** field to enter a password for the second word of the algorithm.

5. Do one of the following:

- If you want to use the Client Activator to customize product-specific activation information for the application, select the **Use Client Activator** check box, then click **Configure** to launch the Activation Wizard.

When you have completed defining your application's activation information, save your project and close the Activation Wizard to return to the SSP Toolkit.

- If you don't want to use the Client Activator, go to the next step.

---

**Tip:** *For more information about the Client Activator, see “How Distributors Activate an Application” on page 247. You may also want to refer to the Client Activator documentation, included in the Client Activator package.*

---

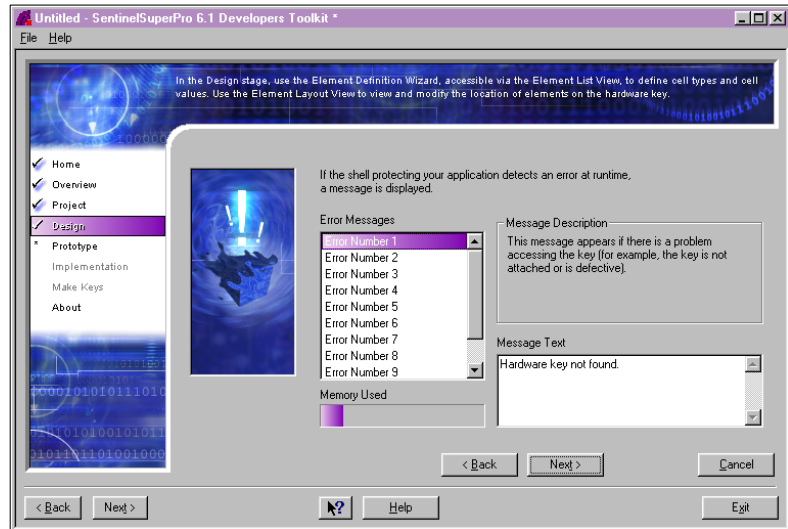
6. Click **Next**.

Element definition is complete and you are returned to the Element List View tab, where your application now appears in the list.

## Customizing Error Messages

If the shell added to your protected application encounters an error at run time, it displays an error message. You can customize these messages.

As you enter text for each message, watch the **Memory Used** status bar. The longer a message is, the more memory it takes. The status bar shows you how much room you have left in the message buffer. To increase the amount of available memory, edit other messages to use fewer characters.



### Customizing Error Messages Screen

To customize an error message:

1. In the **Error Messages** list, select the message you want to customize.

A description of what the message is for and when it appears, displays to the right.

2. In the **Message Text** field, modify the default message text as needed.

The message text is saved automatically.

3. Repeat steps 1 – 2 to edit additional messages.
4. When you have finished customizing your error messages, click **Next** to continue. Element definition is complete and you are returned to the Element List View, where your application now appears in the list.

---

## Protecting Multiple Applications

SentinelSuperPro allows you to protect up to 28 applications on the same key. However, the number of applications is dependent on the number of elements you have programmed on the key. The more elements you have, the more cells that are used, and thus the fewer applications you can protect.

If you want to protect multiple applications on a single key, be sure to also take into consideration how many custom elements you will add. Check the Element Layout View tab to see how many cells you have left on the key for this strategy.

If you want to add more applications to your protection strategy, go back to “Selecting a Protection Type” on page 145.

---

## Editing an Application Protection Element

If you need to make changes to an application protection element, you can edit it at any time. To edit an application protection element:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click on the application you want to edit.
4. Click **Edit**. Each step of the Element Definition Wizard appears as it did when you first created the protected application.
5. Make your changes as necessary, clicking **Next** to move through each step of the wizard.



## Deleting an Application Protection Element

In addition to editing an application element, you can also remove the application from your protection strategy. Be sure you want to remove the application from your protection strategy before you complete the following procedure, as you cannot recover an application once it has been deleted—you would need to add the application again as if it was new.

To delete a protected application:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click on the application you want to delete.
4. Click **Delete**.
5. You are asked if you want to confirm the deletion. Click **Yes**.

The application is deleted from your protection strategy.

---

**Note:** *If you have edited or deleted a protected application **after** you have completed the Prototype stage, you must complete the Prototype stage **again** before you can continue to the Implementation or Make Keys stages.*

---

---

## Where to Go from Here

When you have finished adding all the applications you want to protect as part of this strategy, you are ready to continue to the next step.

To add custom elements, such as algorithms, counters and data words, to your protection strategy, go to Chapter 8, “Working With Design Elements,” on page 169.

Otherwise, go to Chapter 9, “Implementing Your Strategy,” on page 187 to continue.



# Chapter 8

---

## Working With Design Elements

In addition to adding application protection to your protection strategy, you can also add individual design elements, such as algorithms with counters and passwords, individual counters, sublicense limits, and data words that contain data you want to store on the key, such as serial numbers or user data.

These custom elements are added to your strategy through the Element Definition Wizard, similar to how you added application protection to your strategy, making it easy to customize your protection strategy.

---

**Note:** *If you add custom elements, the SSP Toolkit cannot generate the API pseudocode for you. You must write the API calls required to use the values programmed into your keys yourself.*

---

This chapter covers the following topics:

- Types of elements you can add
- Adding algorithms, counters, sublicense limits or data words
- Editing an existing element
- Deleting an element
- Rearranging logical element cell locations

# Custom Element Types

There are several types of custom elements you can add, depending on what you want to do. The following table describes each element.

Custom Element Types

Element Type	Cells	Description	Use If You Want To...
Algorithm	2	A simple algorithm.	Scramble an input string but do not want an associated counter or activation password.
Algorithm with counter	3	An algorithm with an associated counter.	Limit the number of times a demo program can be executed.
Algorithm with password	4	An algorithm that has a password associated with it.	Have the user enter a password to make the application run initially.
Algorithm with counter and password	5	An algorithm that has both a counter and a password associated with it. The algorithm is deactivated when the counter reaches zero. The user must enter a password to reactivate it after the counter reaches zero.	Limit the number of times a demo application can be executed and provide a means for the program to be re-activated in the field.
Algorithm with 2 counters	4	An algorithm that has two counters associated with it.	Use two counters. The first counter that reaches zero deactivates the algorithm, which usually stops your application from executing properly.

**Custom Element Types (Continued)**

Element Type	Cells	Description	Use If You Want To...
Algorithm with 2 counters and 1 password	6	An algorithm that has two counters and a password.	Implement an algorithm that is to be deactivated when either counter reaches zero. The user must enter a password to activate or reactivate the application.
Counter	1	A cell that contains a value you can decrement.	Limit the number of times a demo program can be run, or count the number of times any particular operation is performed.
User data	1	A cell that contains a value your application can test (and change) during execution. If the cell is locked, the value is read-only; your application can read the stored data but cannot change it without the overwrite passwords.	Store a serial number, feature control code or other data you define.
Sublicense	1	A cell that contains a value you select as a sublicense limit.	Restrict the license limit for this application to something less than the hard limit already programmed into the key.

For more information on when to use custom elements, refer to Chapter 4, “Designing Your Protection Strategy,” on page 55.

---

## Adding Algorithms

To add a custom algorithm to your protection strategy:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click **Add**.
4. Select **Custom Element**, then click **Next**.
5. Select the type of algorithm you want to add, then click **Next**.
6. In the **Name** field, enter a name for this algorithm. There is a 16-character limit for element names.

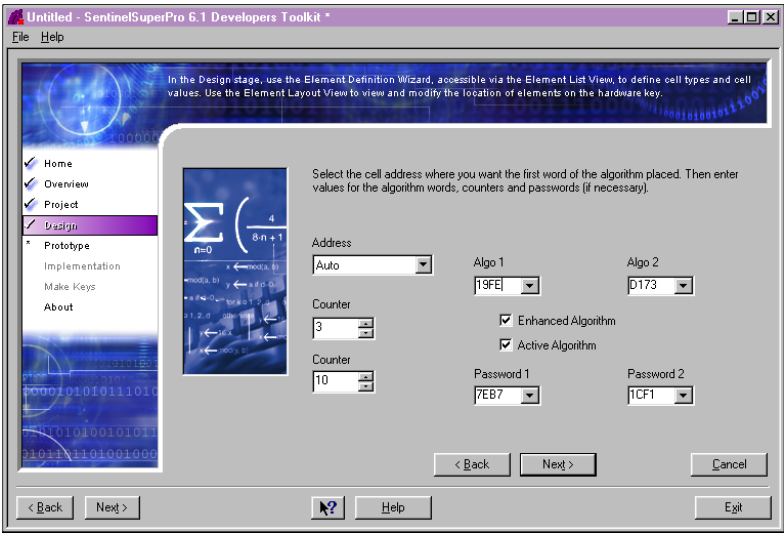
---

**Tip:** *This name will be used throughout the SSP Toolkit to identify this algorithm, so be sure it adequately describes it.*

---

7. In the **Comments** field, enter any additional information about the algorithm you want to save. This field is optional.
8. Click **Next**.

The following screen appears (the number of counters and passwords depends on the type of algorithm you selected in step 5):

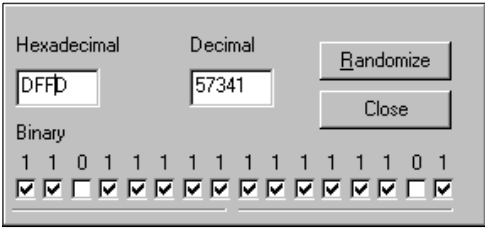


**Options for Algorithm with Two Counters and Two Passwords**

9. From the **Address** drop-down list, select the address of the cell you want the first word of the algorithm to be placed in.

If the location is unimportant to you, select **Auto** to allow the SSP Toolkit to select a location for you.

10. In the **Algo 1** field, click the arrow button to access the Numeric Assistant dialog box.



**Numeric Assistant Dialog Box**

11. Do any of the following, as necessary to generate your value:

- To generate a random value, click **Randomize**.

The Numeric Assistant generates a random value, and provides you with the hexadecimal, decimal and binary equivalents of that value. You can click Randomize as many times as you like.

- To convert a decimal value to a hexadecimal value, enter the value in the **Decimal** field.

The Numeric Assistant automatically calculates the corresponding hexadecimal and binary values.

12. Click **Close**. The value from the Hexadecimal field is transferred to the Algo 1 field.

13. Repeat steps 10 – 12 for the **Algo 2** field.

---

**Note:** *If you already know the value you want to use for the algorithm words, you can enter it directly in the **Algo 1** and **Algo 2** fields. Algo 1 is the first algorithm word and Algo 2 is the second algorithm word.*

---

14. If you want this algorithm to use the enhanced algorithm engine, select the **Enhanced Algorithm** check box.

For maximum security, use of the enhanced algorithm engine is recommended for all algorithms.

15. If you want to make this algorithm active, select the **Active Algorithm** check box. Clear the check box to make the algorithm inactive.

If the algorithm is *active*, the application will be ready to run when shipped to your customer. If the algorithm is *inactive*, the application will not run until it is activated in the field, by the user entering the appropriate activation passwords.



16. Do one of the following:

- If the algorithm you selected has one or more counters associated with it, go to the next section.
- If the algorithm you selected has a password, but no counters, go to “Entering Password Values” on page 176.
- If the algorithm you selected has neither a password, nor a counter associated with it, click **Next**.

Algorithm definition is complete, and you are returned to the Element List View tab, where the algorithm appears in the list.

## Entering Counter Values

If the algorithm you selected has one or more counters associated with it, you need to define the starting values for each counter.

1. In the **Counter** field, enter a number representing the starting value of the counter.

For example, if you want to use the counter to control a demo application’s executions, and you enter **5**, the application can be run five times. The sixth time the user tries to run the application, she will be unable to.

2. Repeat step 1 for the second counter, if applicable.

3. Do one of the following:

- If the algorithm you selected also has a password, go to “Entering Password Values” below.
- If the algorithm you selected does not have a password associated with it, click **Next**.

Algorithm definition is complete, and you are returned to the Element List View tab, where the algorithm appears in the list.

## Entering Password Values

If the algorithm you selected has a password associated with it, you need to define the password values for each word of the algorithm.

1. In the **Password 1** field, click the arrow button to access the Numeric Assistant dialog box, and enter an activation password for the first word of the algorithm.

See page 173 for instructions on using the Numeric Assistant dialog box.

2. Repeat step 1 for the **Password 2** field to enter a password for the second word of the algorithm.
3. Click **Next**.

Algorithm definition is complete, and you are returned to the Element List View tab, where the algorithm appears in the list.

---

## Adding Counters

To add a single counter to your protection strategy:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click **Add**.
4. Select **Custom Element**, then click **Next**.
5. Select **Counter**, then click **Next**.
6. In the **Name** field, enter a name for this counter. There is a 16-character limit on the counter name.

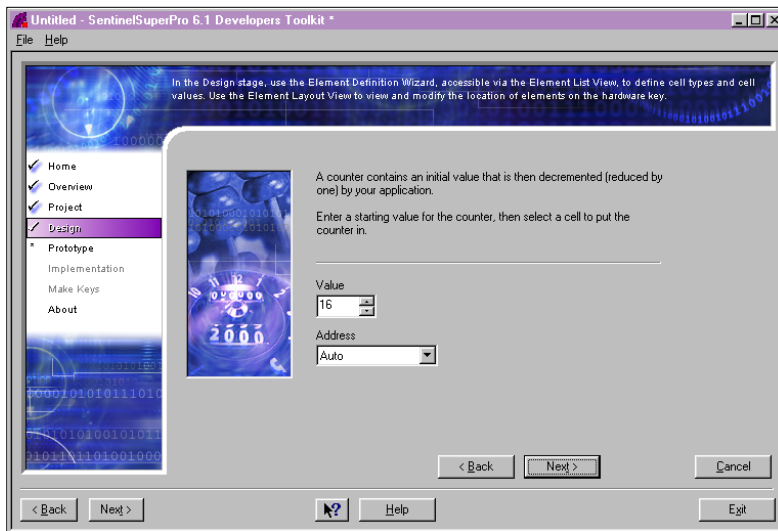
---

**Tip:** This name will be used throughout the SSP Toolkit to identify this counter, so be sure it adequately describes it.

---

7. In the **Comments** field, enter any additional information about the counter you want to save. This field is optional.
8. Click **Next**.

The following screen appears:



### Custom Element Counter Options

9. In the **Value** field, enter a number representing the starting value of the counter.
10. From the **Address** drop-down list, select the address of the cell you want the counter to be placed in.  
  
If the location is unimportant to you, select **Auto** to allow the SSP Toolkit to select a location for you.
11. Click **Next**.

Counter definition is complete and you are returned to the Element List View tab, where the counter appears in the list.

---

## Adding Data Words

To add a data word to your protection strategy:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click **Add**.
4. Select **Custom Element**, then click **Next**.
5. Select **User Data**, then click **Next**.
6. In the **Name** field, enter a name for this data word. There is a 16 character limit on the data word name.

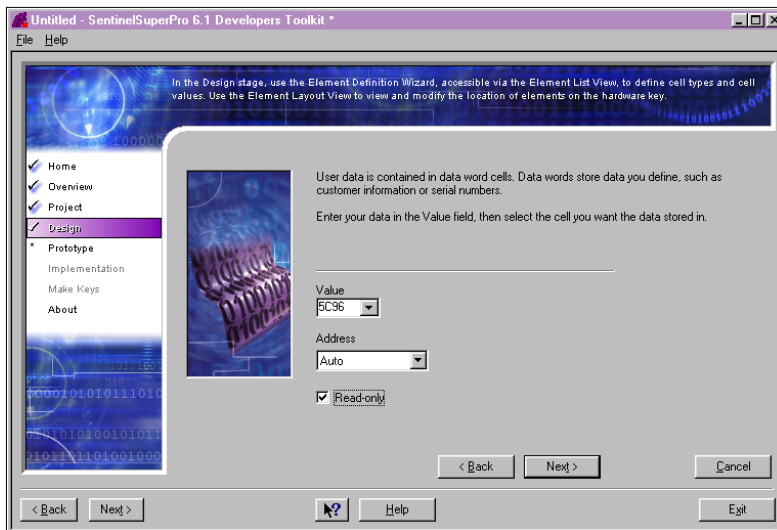
---

**Tip:** *This name will be used throughout the SSP Toolkit to identify this user data cell, so be sure it adequately describes it.*

---

7. In the **Comments** field, enter any additional information about the data you want to save. This field is optional.
8. Click **Next**.

The following screen appears:



### Custom Element User Data Options

9. In the **Value** field, enter the data you want stored in this cell.

The maximum size of the stored data is 16 bits (four hex characters). You can use a 16-bit word to represent letters; how you encode them determines how many you can use.

10. From the **Address** drop-down list, select the address of the cell you want the data to be placed in.

If the location is unimportant to you, select **Auto** to allow the SSP Toolkit to select a location for you.

11. If you want your application to be able to read this data, but *not* make changes to it, select the **Read-Only** check box.

If you leave this check box blank, your application will be able to change the data located in this cell.

12. Click **Next**.

Data word definition is complete and you are returned to the Element List View tab, where the data word element appears in the list.

---

## Adding Sublicense Limits

To add a sublicense limit to your protection strategy:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click **Add**.
4. Select **Custom Element**, then click **Next**.
5. Select **Sublicense**, then click **Next**.
6. In the **Name** field, enter a name for this sublicense. There is a 16 character limit on the sublicense name.

---

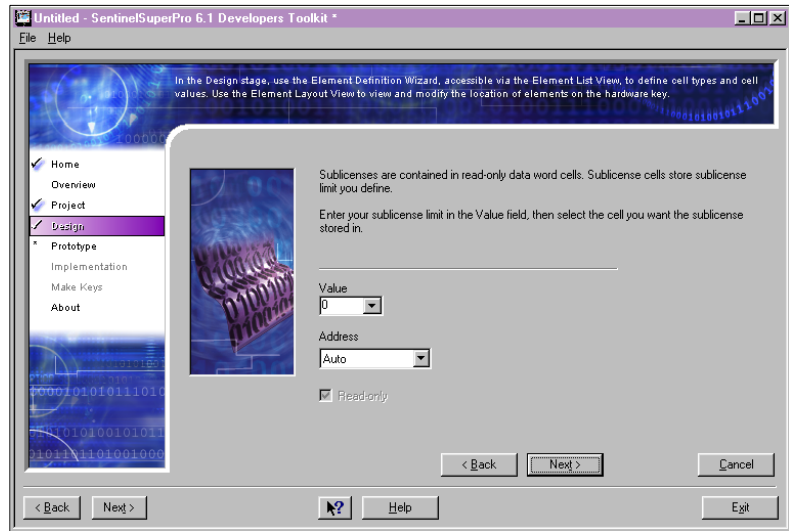
**Tip:** *This name will be used throughout the SSP Toolkit to identify this sublicense, so be sure it adequately describes it.*

---

7. In the **Comments** field, enter any additional information about the sublicense you want to save. This field is optional.
8. Click **Next**.



The following screen appears:



### Custom Element Sublicense Options

9. In the **Value** field, enter the sublicense value.

The sublicense value should be less than the hard limit on the keys you will be using with your application. For example, if the hard limit is 25, the sublicense value should be 24 or less.

10. From the **Address** drop-down list, select the address of the cell you want the sublicense to be placed in.

If the location is unimportant to you, select **Auto** to allow the SSP Toolkit to select a location for you.

Note that Sublicense cells are read-only by default.

11. Click **Next**.

Sublicense definition is complete and you are returned to the Element List View tab, where the sublicense element appears in the list.

## Editing Existing Elements

If you need to make changes to your custom elements, you can edit them at any time. To edit a custom element:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click on the element you want to edit.
4. Click **Edit**. Each step of the Element Definition Wizard appears as it did when you first created the element.
5. Make your changes as necessary, clicking **Next** to move through each step of the wizard.

## Deleting an Element

In addition to editing an element, you can also remove the element from your protection strategy. Be sure you want to remove the element from your protection strategy before you complete the following procedure, as you cannot recover an element once it has been deleted—you would need to add the element again as if it was new.

To delete an element:

1. Navigate to the **Design** stage.
2. Verify you are on the **Element List View** tab.
3. Click on the element you want to delete.
4. Click **Delete**.
5. You are asked if you want to confirm the deletion. Click **Yes**.

The element is deleted from your protection strategy.

---

**Note:** *If you have edited or deleted an element **after** you have completed the Prototype stage, you must complete the Prototype stage **again** before you can continue to the Implementation or Make Keys stages.*

---

---

## Rearranging Elements on the Key

Once you have defined all the elements in your protection strategy, including application protection elements, you can view where the element cells are located on the key. You can also rearrange the elements in your strategy using the drag-and-drop method.

For example, you may want to rearrange elements if an element must be in a certain cell, or if you are increasing the number of applications you are protecting with this key and need to make room for the additional applications.

---

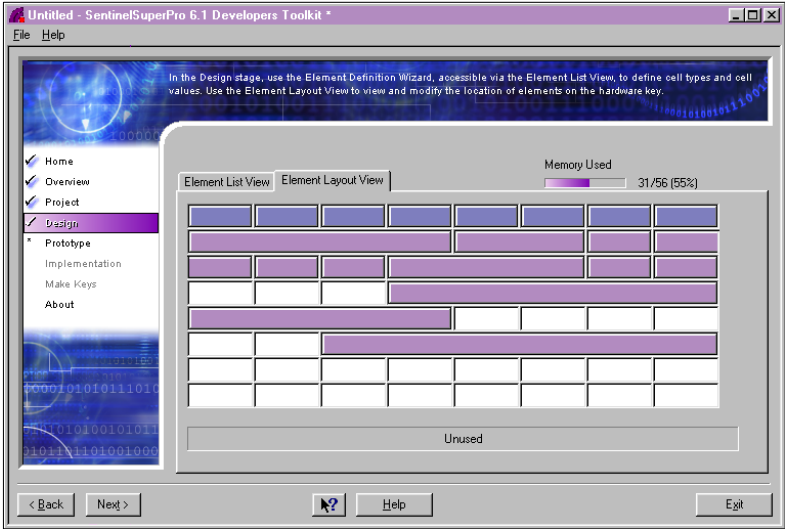
**Note:** *If you selected the One Time Update option in the Developer Configuration dialog box, an element for the update will appear on the key. You cannot move this element via the drag-and-drop method. For more information about this option, see page 112.*

---

To view and/or move the location of elements on the key:

1. Navigate to the **Design** stage.
2. Click the **Element Layout View** tab.

The following screen appears:



**Element Layout View Tab**

- 3. Move your cursor over the cells to view the name of the element located in the cell.
- 4. Click and drag an element to move it to a new location on the key.

You cannot move an element to an illegal location on the key. Review “Valid Algorithm Addresses” on page 47 for more information on valid addresses for algorithms and counters.

---

**Warning!** *If you move a counter to the immediate left of an algorithm, the counter will function as an algorithm counter. When the counter reaches zero, the algorithm will be deactivated, even if you did not intend for that to happen.*

---

# Chapter 9

---

## Implementing Your Strategy

When you have finished designing your protection strategy—all your design elements are in place and you have defined what type of application protection to use—you are ready to implement your strategy.

The first step in implementing your strategy is to create a prototype hardware key. During the prototyping, the SSP Toolkit writes your protection strategy elements to the key, generates query/response pairs, adds default field activation actions, and generates pseudocode.

The second (optional) step is to verify the key was programmed correctly by viewing the cell values in MemView.

The last step in implementing your strategy is to add the appropriate API function calls to your source code, either manually if you are using integrated protection, or by “shelling” your application if you selected automatic protection.

This chapter covers the following topics:

- Creating the prototype
- Verifying the key using MemView
- Shelling an application
- Adding API functions to your source code

---

## Creating the Prototype

When you create a prototype of your protection strategy, you are actually programming a master key with all of the elements you previously defined as part of your protection strategy.

Only those cells containing an element are overwritten; the more elements you have in your strategy, the longer the prototype process takes. Any cells that are not used in your strategy, but were previously programmed are left alone.

---

**Note:** *If you delete an element from your strategy, and then repeat the prototype process, when you view the key using MemView, it may look like the element is still there. This is because when you prototype a key, it skips any unassigned cells. Since the cells from the element you deleted are now unassigned, they weren't overwritten, and thus still contain the values from the previous prototype. Despite this, the element **has** been deleted and is not programmed into the key.*

---

During prototyping, after the key is programmed, SentinelSuperPro also performs the following functions:

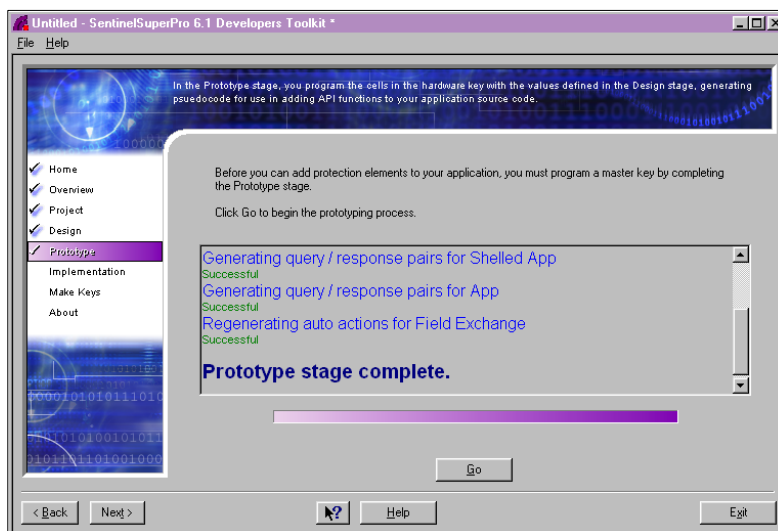
- **Generates query/response pairs for each application included in your strategy.** Queries are used to verify the presence of the key while your application is running. Random query values—and lots of them—in your code makes your application more secure. You can view the query/response pairs in the pseudocode; see “Viewing the Pseudocode” on page 192.
- **Defines default field activation actions and commands.** If any of the applications or custom element algorithms in your strategy use the activation type Static, Trusted or Distributed, SentinelSuperPro creates a default action and command for activating the application in the field. See “Working with Actions” on page 202 for more information.
- **Generates a pseudocode protection plan.** The pseudocode protection plan outlines the API functions you need to add to your

application (if you are using integrated protection), as well as additional information about your protection strategy. See “Adding API Functions to Your Source Code” on page 192 for more information.

---

**Note:** *The Prototype stage is a required stage in the SSP Toolkit—when you create a prototype, you are committing to your strategy’s design. If, after creating an initial prototype, you return to the Design stage to edit or add more elements, you must repeat the prototyping process before you can continue to the Implementation and Make Keys stages. The Prototype stage may be repeated as many times as necessary.*

---



**Prototype Stage with Status Messages**

## Starting the Prototype Process

1. Navigate to the **Prototype** stage.
2. Click **Go**.

The prototype process begins. The status and outcome of each stage of the process appears in the text box. The more elements you have in your strategy, the longer the prototype process takes. When the “Prototype Stage Complete” message appears, you are ready to continue.

---

**Note:** *If a message appears informing you that your developer ID or passwords are incorrect, you need to return to the Project stage and enter the correct ID and/or passwords before you can continue. See “Changing Your Developer ID or Passwords” on page 133 for more information.*

---

3. Go to the next section if you want to verify the key was programmed correctly (this is an optional step).

---

## Verifying the Key Using MemView

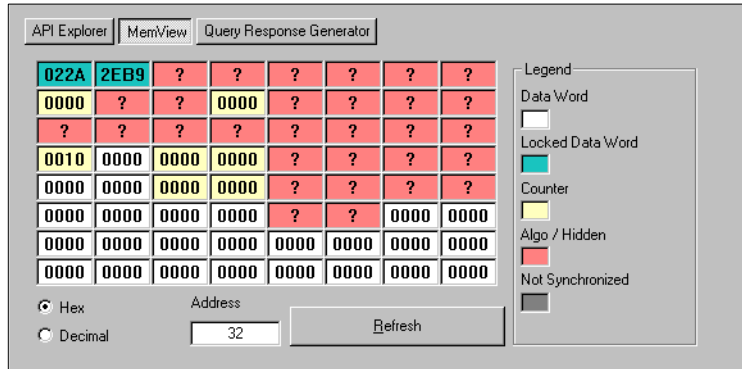
After you have successfully programmed your first key in the Prototype stage, you may want to use the MemView section of the API Explorer to verify that the key was programmed accurately (for example, your algorithms are in the appropriate cells) and that your protection strategy is correct.

If you find something is missing (for example, you forgot to add a counter or data word), or you want to change an existing element in your strategy, return to the Design stage to make your changes, then repeat the prototyping process.

To verify the key using MemView:

1. Navigate to the **Implementation** stage.
2. Click the **API Explorer** tab.
3. Click **MemView**. The MemView section appears with all cells shaded grey. This means the key has not yet been queried for the status of the programmed cells.
4. Click **Refresh**. The SSP Toolkit queries the key, returning the access code and value of each cell.





### MemView – Programmed Key with Counters and Algorithms

Cell access codes are identified by different colors, as shown in the legend on the right. Cell values are provided in hexadecimal format within each individual cell. A question mark (?) represents a restricted cell, or a cell programmed with an algorithm value.

To view a selected cell's address, move the mouse pointer over that cell. The address appears in the Address field automatically. To view cell values and addresses in decimal format, select **Decimal** in the lower left of the window.

After you have verified your key was programmed correctly and your protection strategy is complete, you are ready to add protection to your application code.

- If you are using *both* integrated and automatic protection for a single application, you must first add the appropriate API function calls to your source code and then recompile. After you have recompiled, you can apply the shell to the executable file. Go to the next section to add the API function calls.
- If you selected only integrated protection for your application(s), you need to add API function calls directly to your source code. Go to the next section.
- If you selected only automatic protection for your application(s), go to “Shelling an Application” on page 196.

---

## Adding API Functions to Your Source Code

If you are using integrated protection for your application, you must manually add the appropriate API function calls to your application's source code in order to implement your protection strategy. The SSP Toolkit provides you with pseudocode that tells you what functions you need to add to your code.

---

**Note:** *If you are using **only** automatic (“shelled”) protection for your application, the appropriate API functions are added during the shell process. You do not need to manually add API function calls to your application source code. Go to “Shelling an Application” on page 196.*

---

### Viewing the Pseudocode

The pseudocode protection plan generated by the SSP Toolkit outlines your protection strategy. You can display this information within the Implementation stage, or you can save it as a file to your hard drive, where you can then open it in any text editor to view, edit or print out. Pseudocode is provided for the most frequently used development languages: ANSI C, Visual Basic and Pascal.

---

---

**Warning!** *The pseudocode protection plan contains your developer ID and passwords—make sure you keep it secure!*

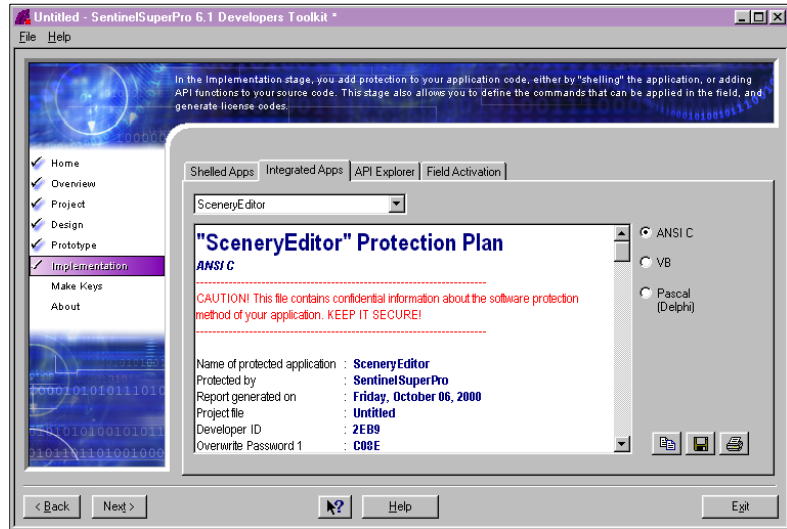
---

---




To view the pseudocode for your development language:

1. Navigate to the **Implementation** stage.
2. Click the **Integrated Apps** tab.
3. From the drop-down list, select the application you want to view the pseudocode protection plan for.
4. From the options on the right, select the development language you want to view the pseudocode in.

The pseudocode protection plan for the selected application and language appears in the text box.



### Implementation Stage – Pseudocode Protection Plan

5. To save the plan to a file, click the save button .
6. To print the plan, click the print button .
7. To copy the plan to the clipboard, click the copy button .

The first part of the plan describes what you need to do in your development language to prepare to use the SentinelSuperPro API.

The next part of the plan shows you the expected responses for queries to the algorithms programmed into your keys. Random query strings and their corresponding response strings (generated during the Prototype stage) are included, even for applications that are designated as inactive. You may wish to copy this section from the plan and paste it into your source code.

The query section is followed by pseudocode for the API function calls required to implement your protection strategy.

### Adding Code to Your Application

To add API function calls to your application code, look at the example code provided for your development language. This code is available from the SentinelSuperPro installation Web site. See “Installing SentinelSuperPro Interfaces” on page 31 for more information.

The example file shows the exact syntax for each SentinelSuperPro API function. Referring to the example, add the function calls specified in the pseudocode file to your application’s source code.

---

**Tip:** For more information about SentinelSuperPro API functions, please refer to Chapter 15, “API Function Reference,” on page 281.

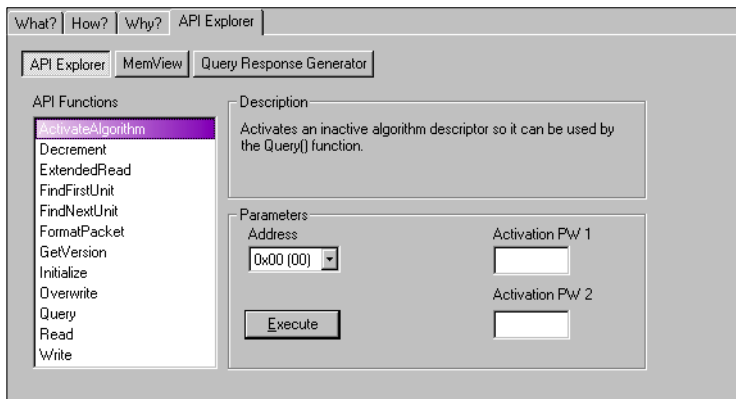
---

### Using the API Explorer to Evaluate the API Functions

If you want to see expected responses, or evaluate the behavior of the API functions with your programmed key prior to putting them in your source code, use the API Explorer.

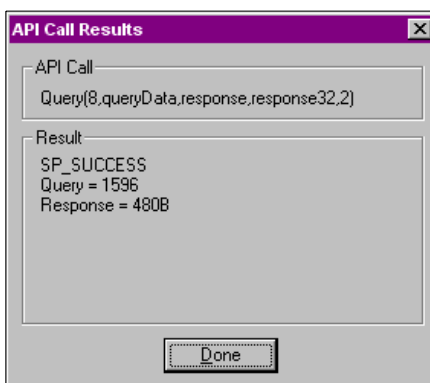
To test an API function on a selected cell:

1. Navigate to the **Implementation** stage.
2. Click the **API Explorer** tab.
3. If necessary, click **API Explorer** to access the API Explorer section.



**API Explorer**

4. From the API function list, select a function. A description of the function appears to the right.
5. Under **Parameters**, select values for the available parameters.  
See page 124 for more detailed information about selecting parameters.
6. Click **Execute**. The API function is invoked, with the parameters you selected, and the API Call Results message box appears:



**API Call Results Message Box**

7. Click **Done** to close the message box.
8. Repeat steps 3 through 6 to test additional API functions.

---

**Note:** Remember, you need to call the `RNBOSproFormatPacket()` and `RNBOSproInitialize()` functions prior to calling any other function.

---

---

## Shelling an Application

Applications you have defined as using automatic (“shelled”) protection are easy to implement, as all you need to do is click a button to add the protective shell layer to your executable file. Once the shell has been added, all the protection options you defined for the application are in place.

---

**Note:** If you are shelling an application that you also added integrated protection to, you must add the API function calls to your source code first and then recompile **before** you add the shell.

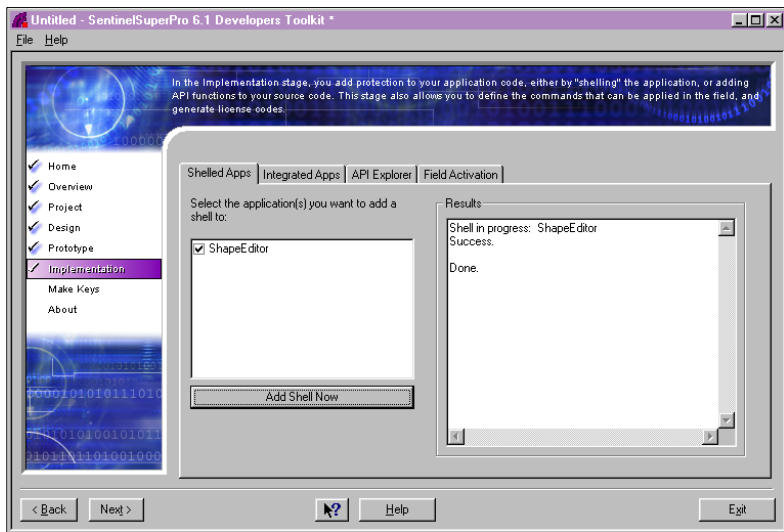
---

To add the shell to an application:

1. Navigate to the **Implementation** stage.
2. Click the **Shelled Apps** tab.
3. In the **Shelled Applications** list, select the application(s) you want to add a shell to.

Only those applications you defined in your protection strategy as using the automatic protection type are available in the list.

4. Click **Add Shell**. The shell layer is added to the application's executable file. You can view the status of the shell operation in the Results text box.



### Implementation Stage – Shelling an Application

5. Go to the next section to continue.

## Testing Your Application Protection

At this point, you have defined your protection strategy, programmed a prototype hardware key, and added the appropriate code—either manually or via a shell—to protect your application.

Before you continue, we recommend testing your application to verify that it executes correctly with the appropriate hardware key both attached and missing. To do so:

1. Make sure the SentinelSuperPro hardware key is attached to your computer.
2. Verify the SentinelSuperPro server is running on your system.
3. Execute your protected application.

With the key attached, the application should run normally.

4. Exit the application.
5. Remove the SentinelSuperPro key from your computer.
6. Execute your protected application.

With the key missing, you should be unable to run your application and an error message should appear.

If you have designed your application to be a network application, you may also want to test that your application can find a key located on another computer on the network. To do so:

1. Connect the SentinelSuperPro hardware key to another computer on the same network as your computer.
2. Start the SentinelSuperPro server on the system you connected the key to.



3. On your computer, execute your protected application.

With the key attached to the server, the application should be able to obtain a license and run normally.

4. Use the application for several minutes, to verify that heartbeat messages are being sent appropriately and that you do not receive any time-out errors from the server.
5. With the application still running, go to the server computer and open the SentinelSuperPro Monitoring Tool to verify that the key is showing a single license in use.
6. Exit the application.
7. Remove the SentinelSuperPro key from the server.
8. On your computer, execute your protected application again.

With the key missing from the server, you should be unable to obtain a license and an error message should appear.

If your protected application does not respond as expected in either the standalone or the network scenarios, review your protection strategy for missing elements or errors by rereading Chapters 7 and 8 in this guide. If, after reviewing your strategy, you still need assistance, please contact Rainbow Technologies Technical Support. See “Getting Help” on page xxiii for Technical Support contact information.



# Chapter 10

---

## Defining Field Activation Actions

To be able to update keys in the field, you need to define the field activation actions and commands that can be performed on those keys within your protection strategy.

Commands are API function calls that describe what will be done to the key in the field. For example, the Decrement Counter command locates the counter cell on the key and decrements it by the value you specify. Actions are groups of one or more commands.

When you generate a license code, the actions and commands you select are encrypted into the license code specific to the selected key. When the license code is entered in the Client Activator or Field Exchange Utility, the actions and commands are applied to the key.

This chapter includes the following topics:

- Adding actions
- Adding commands
- Available commands
- Testing your strategy

---

## Working with Actions

Actions are groups of commands. This allows you to group a set of commands together so you—or your distributors—don't have to select the commands individually when generating license codes for field updates. This is especially helpful for those people who will be generating license codes, but don't need to be exposed to the complexities of the commands being used.

Actions and commands for activating an inactive application (activation type of Static, Trusted or Distributed) are added automatically during the Prototype stage. These actions and commands cannot be changed or deleted, as they are the default means of activating your application.

Automatically added actions and commands can be identified with red asterisks (\*) on their icons.

---

**Note:** *If you selected the Distributed activation type, be sure to add an action and command for incrementing the counter on the distributor key. This will allow you to add more licenses to your distributor's key in the field.*

*When you create a .DST file for your distributor, the command for incrementing the distributor key counter is NOT included. This prevents your distributor from giving himself more update licenses without your approval.*

---

If you change your strategy by returning to the Design stage, these actions and commands are appropriately updated or removed automatically during the Prototype stage.

You must define actions before you can add commands to your protection strategy.

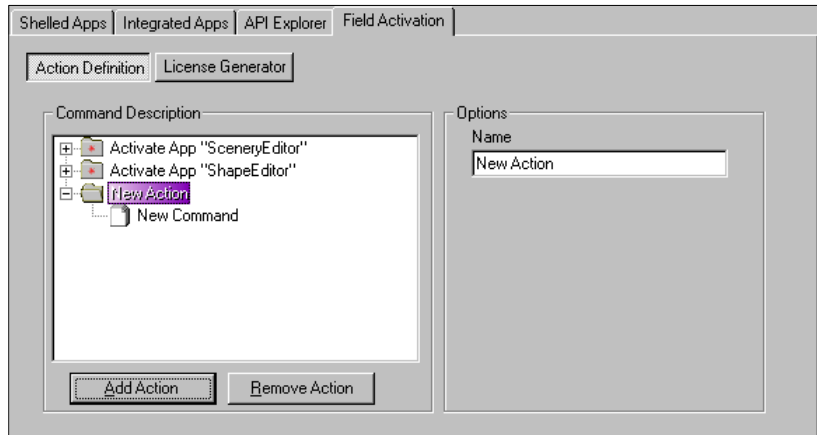
## Adding an Action

1. Navigate to the **Implementation** stage.
2. Click the **Field Activation** tab. The Field Activation section appears with the Action Definition window open.
3. Click **Add Action**.

A new action appears in the Command Description list.

4. Under **Options**, in the **Name** field, enter a name for the action.

The name should be concise, yet descriptive, so the people generating license codes can easily see how the key will be updated if they include this action in the license code update script.



**Action Definition – Adding an Action**

## Removing an Action

If you no longer want an action in your protection strategy, or you have made a mistake, you can delete actions. To do so:

- Select the action you want to delete, then click **Remove Action**.

The action—and all its corresponding commands—is removed from the Command Description list.

---

**Note:** *You are not asked to confirm the action deletion. Be sure you have selected the right action, and that you want to delete this action permanently, before you click Remove Action.*

---

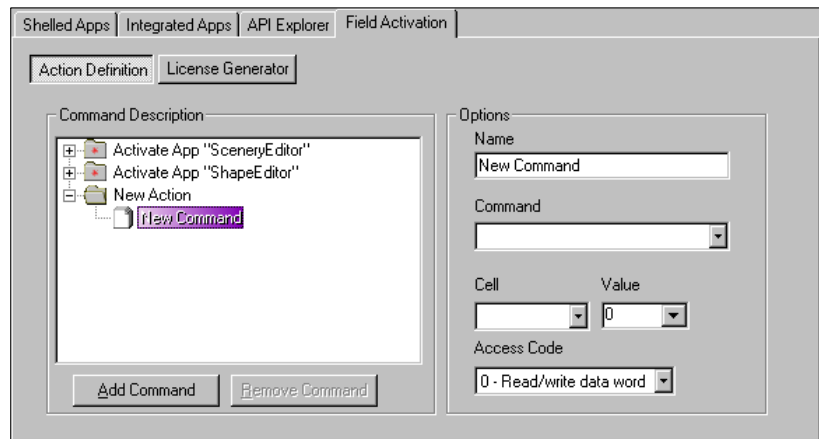
If you have provided your SentinelSuperPro project file to users who are generating license codes, you need to send a new file to those users after you have removed an action. This prevents them from selecting the action you just removed to update a key in the field.

## Working with Commands

Once you have defined your actions, you must add commands.

### Adding a Command

1. Navigate to the **Implementation** stage.
2. Click the **Field Activation** tab. The Field Activation section appears with the Action Definition window open.
3. Select an action. If no commands are listed under the action, expand the action by clicking the **+** to the left of the action name.



#### Action Definition – Adding a Command

4. Click on any command (except an auto-generated command), then click **Add Command**. A new command icon appears in the action tree.
5. Under **Options**, in the **Name** field, enter a name for the command.

The name should be concise, yet describe what the command will do to the key in the field.

6. From the **Command** drop-down list, select the command you want to perform on the key.

See “Available Commands” on page 209 for a list of available commands and descriptions of each.

7. From the **Cell** drop-down list, select the address of the cell you want the command performed on.

---

**Note:** *To view the cells being used in your protection strategy (and the elements occupying them), open the **Design** stage, then click the **Element Layout View** tab. See “Rearranging Elements on the Key” on page 185 for more information. To view the addresses of these cells, open the **Implementation** stage, click the **API Explorer** tab, then click **MemView**. See “Viewing Memory Cells” on page 127 for more information.*

---

8. In the **Value** field, enter the value you want written to the selected cell.

Click the arrow to access the Numeric Assistant dialog box. See page 173 for instructions on using the Numeric Assistant.

Use the table on the next page as a guide for what values to enter, based on the command you have selected.

---

---

**Warning!** *If you chose not to include the overwrite passwords in the field exchange DLLs, you should not use commands that will write to a cell, such as Write Cell or Increment Counter. This is because the overwrite passwords are required in order to write to a cell; without the overwrite passwords, the write will not occur, your user will see an error, and the key will not be updated. See “Including Overwrite Passwords in the Field Exchange DLLs” on page 115 .*

---

---



## Field Activation Commands and Values

Selected Command	Description of Value
<b>Write Cell</b>	The value you want written to the cell. Enter your own value, or use the Numeric Assistant to generate a random value.
<b>Activate Algo PW1</b>	The first word of the algorithm's activation password. Points to the first word of the algorithm. See "Using Activation Passwords" on page 76 for more information.
<b>Activate Algo PW2</b>	The second word of the algorithm's activation password. Points to the first word of the algorithm. See "Using Activation Passwords" on page 76 for more information.
<b>Decrement Counter</b>	The selected cell is decremented by one.
<b>Increment Distributor Counter</b>	The value you want the activation counter on a distributor key incremented by. This command is available only if you have a distributed application included in your protection strategy.
<b>Increment Counter</b>	The value you want the selected cell incremented by.
<b>Bit Mask AND</b>	Value you enter is ANDed to the value in the cell.
<b>Bit Mask OR</b>	Value you enter is ORed with the value in the cell.
<b>Decrement Counter to Zero</b>	The Value field does not appear when you select this command. Because it decrements a counter cell to zero, no value is necessary.

9. If it is displayed, from the **Access Code** drop-down list, select the access code you want assigned to the selected cell.

The access code determines how you want to use the selected cell. For example, to make the cell an algorithm cell, select access code 3; to make the cell a counter, select access code 2. See “Cell Types” on page 37 for more information.

### Removing a Command

If you no longer want a command in your protection strategy, or you have made a mistake, you can delete commands, just as you can actions.

---

**Tip:** To delete all commands in an action, it may be easier to delete the action itself—see “Removing an Action” on page 204 for more information.

---

To remove a command:

- Select the command you want to delete, then click **Remove Command**.

The command is removed from the Command Description list.

---

**Note:** You are not asked to confirm the command deletion. Be sure you have selected the right command, and that you want to delete this command permanently, before you click Remove Command.

---

Because each action must have at least one command, you cannot delete a command if it is the only command listed under the selected action. If you need to delete such a command, click **Add Command** first to add a new, undefined command, then remove the first command as described above.

If you have provided your SentinelSuperPro project file to users who are generating license codes, you need to send a new file to those users after you have removed a command. This prevents them from selecting the command you just removed to update a key in the field.

## Available Commands

The following commands are available for field activation using SentinelSuperPro keys:

Command	Description
<b>Write Cell</b>	Writes the value you entered to the selected cell.
<b>Activate Algo PW1</b>	Enables an inactive algorithm already on the key. The value you enter is passed as a parameter to the Activate function. This command must be used in conjunction with the Activate Algo PW2 command. See "Using Activation Passwords" on page 76.
<b>Activate Algo PW2</b>	Enables an inactive algorithm already on the key. The value you enter is passed as a parameter to the Activate function. This command must be used in conjunction with the Activate Algo PW1 command. See "Using Activation Passwords" on page 76.
<b>Decrement Counter</b>	Decrements a counter cell. This command reads the current counter value and then subtracts one from the value.
<b>Increment Distributor Counter</b>	Increments the activation counter cell on a distributor key. This command reads the current counter value and then adds the value you specified. This command is available only if you have a distributed application included in your protection strategy.
<b>Increment Counter</b>	Increments a counter cell. This command reads the current counter value and then adds the value you specified.
<b>Bit Mask AND</b>	Used to clear a bit in a cell value.
<b>Bit Mask OR</b>	Used to set a bit in a cell value.
<b>Decrement Counter to Zero</b>	Decrements a counter cell to zero, regardless of the current value. Typically used when you are updating a demo to a fully-licensed version; must be used in conjunction with other commands. See "Controlling Demo Applications" on page 90.

## Testing Your Strategy

After you have defined all your actions and commands for field activation, you may want to test your strategy before you start manufacturing keys and shipping your application to verify that the commands are applied appropriately and the key is correctly updated.

To test your protection strategy's field activation commands:

1. Program a key using your protection strategy. See Chapter 11, “Programming Keys,” on page 213 for instructions.
2. Connect the key to any workstation and use it to run your application.
3. Open the Client Activator or the Field Exchange Utility on the workstation you ran your application on, and generate a locking code.
4. On a different workstation, start the SSP Toolkit, open the **Implementation** stage, and then click on **License Generator**.

You can also use the License Generator Utility, if desired.

5. Enter the locking code and generate a license code, selecting the actions you want to test. See Chapter 13, “Activating and Updating Keys,” on page 243 for instructions.
6. On the workstation running your application, enter the license code you generated in step 5 in the Client Activator or the Field Exchange Utility.
7. After the license code updates the key, check to see if you have access to the expected areas of your application, based on the action(s) you selected in step 5.
8. Remove the key you updated from the workstation.

9. Connect the key to the workstation you are running SentinelSuperPro on to view the cells and their values using **MemView** in the **API Explorer** section of the **Implementation** stage.

The values should match those you entered when you were defining the commands. See “Viewing Memory Cells” on page 127 for more information about using MemView.

---

## Final Steps

Now that your strategy is complete, the final step in preparing the application for shipping is to compile it.

Compile the application and link it with the appropriate interface modules and the Sentinel driver. Use the readme file for your development language to determine what file(s) you need.

Once linked, the interface module and Sentinel driver handle all communication between your application and the key.

---

**Note:** *The above steps are for applications using integrated protection only. If you are using automatic (shelled) protection, you do not need to compile or link your application. A shelled application is ready to ship as soon as you have applied the shell.*

---



# Chapter 11

---

## Programming Keys

Once you have completed your protection strategy, including prototyping a master key, you are ready to start programming product keys to include in the final package with your protected application.

Typically, as a developer, you will not be responsible for programming the keys that will be shipped to your customers. This task is usually handled by your company's manufacturing department as part of the assembly-line process for creating your product.

To avoid giving your manufacturing department access to your passwords—which would also give them the ability to change field activation commands or other elements in your protection strategy—we recommend providing them with the Make Keys Utility included with SentinelSuperPro. For more information about this utility, please refer to Chapter 14, “Using the Stand-alone Utilities,” on page 259.

You may also decide to allow your distributors to program product keys. In this case, you will need to provide those distributors with a pre-programmed distributor key to meter the number of keys they activate and/or update. You may be responsible for programming distributor keys, or this task may also be handled by your company's manufacturing department.

The instructions in this chapter are provided so you can familiarize yourself with the key programming procedures used by your manufacturing department, or so you can program product or distributor keys yourself if necessary.

This chapter covers the following topics:

- Setting up to program product keys
- Programming a product key
- Programming a distributor key



---

## Setting Up to Program Product Keys

Before you begin programming product keys, make sure you have an adequate stock of keys with your assigned developer ID. The keys you will be programming as product keys *must* have the same developer ID as the key used while you designed your protection strategy.

If you need additional keys, please contact your Rainbow Technologies sales representative or distributor.

### Selecting the Appropriate Keys

Depending on what you have implemented in your protection strategy, you will be programming either stand-alone keys or network keys.

Network keys, which are pre-programmed by Rainbow with the hard limit you specified when you placed your order, can be identified by the words “SuperPro Net” pressed into the plastic on the key.

It is important that you select the appropriate key for programming. Stand-alone keys allow only one user per key, even when used in a network environment. Programming stand-alone keys when you should be using network keys will decrease the number of licenses available to your customers.

### Connecting the Keys

As you program multiple keys, you will be taking SentinelSuperPro keys on and off your workstation frequently. As a result, you will want to protect connectors from being damaged, while at the same time making it convenient to connect and disconnect keys.

---

**Note:** *You cannot cascade keys while they are being programed. Only one key can be attached at a time during programming.*

---

To solve both issues, we recommend attaching a shielded cable with appropriate connectors for the key you are programming that reaches from the

port to a work surface, preventing you from having to reach or bend over to connect or disconnect keys.

Do not remove a key while it is being programmed, as a write failure will occur. A key removed during programming can be reprogrammed.

### ***Using Cables with the SentinelSuperPro Hardware Key***

Due to the large variety of cables currently on the market, Rainbow Technologies does not recommend a specific brand or type of cable for use with the SentinelSuperPro key, nor do we guarantee that all cables will be compatible with the key.

However, we do recommend the following:

- Cables should not be longer than 6 feet in length.
- Cables should be shielded.
- Do not use ribbon cables.
- Cables must be straight-through; that is, they must have all pin signals wired through to the connectors on either end of the cable.

Please be aware that cable connectors may only be used for a specific number of times, perhaps for as few as 100 connections. Contact the manufacturer of your cable/connector for specific information on how many insertions the cable/connector is rated for. Based on that specification, change the cable/connector on a regular basis, as needed.

Additionally, since the insertion life of the connector on the computer you are using for programming is also limited, you may want to consider using Rainbow Technologies' key programming services as a solution for reliable, high-volume key programming.

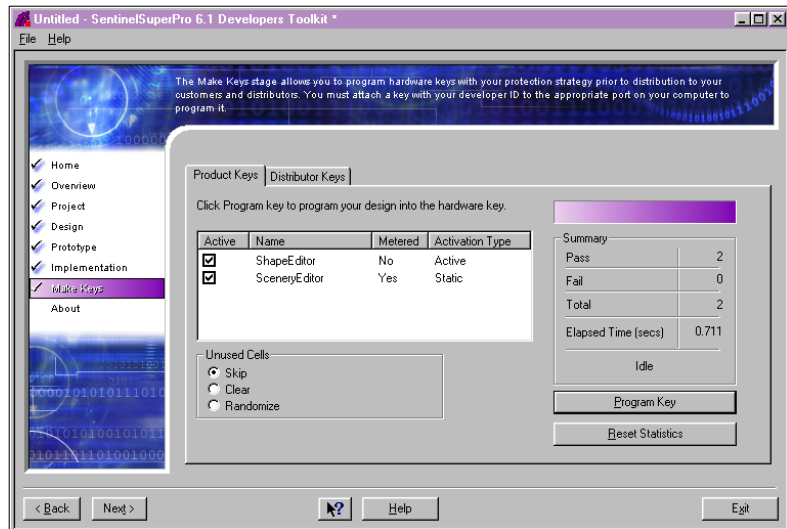
## Programming a Product Key

To program a product key:

1. In the SSP Toolkit, open the SentinelSuperPro project containing the protection strategy for the application you are programming a key for.
2. Navigate to the **Make Keys** stage, and verify you are on the Product Keys tab.

**Note:** If you did not complete the Prototype stage, the Make Keys stage is unavailable and you are not ready to program product keys. Refer to Chapter 9, “Implementing Your Strategy,” on page 187 for more information on completing the Prototype stage.

A list of the applications you applied integrated or automatic protection to appears. The values under **Metered** indicates whether or not the application has a demo counter or metering options associated with it.



**Make Keys Stage – Product Keys Tab**

3. Connect the key you want to program to the appropriate port on your workstation.
4. To override the activation status of an application for this key only, select the **Active** check box for the appropriate application.

The default activation status for each application depends on the activation type you selected during the Design stage. You may want to change this status for selected keys only—this option allows you to do so on a per key basis without changing your overall strategy.

For example, you can make a demo application inactive so that it must be activated in the field.

If the **Active** check box is selected, the application will be shipped as active for this key only. If the check box is cleared, the application will be inactive upon shipment, requiring the user to enter an activation password to run the application.

---

**Note:** *Because the override is on a per key basis, if you make a change to the activation status, and then go to another stage, when you return to the Make Keys stage the status change is **not** saved.*

---

5. Under **Unused Cells**, select one of the following:
  - **Skip:** Overwrites any cells used in your protection strategy, but leaves any cells that are unallocated or not assigned value in your protection strategy. This is the default option.
  - **Clear:** Removes values and access codes/cell types from all cells not used in your protection strategy, including any previously programmed cells. Both the cell value and access code is set to 0.
  - **Randomize:** Randomly assigns values and access codes to cells not used in your protection strategy. This is the most secure option; it makes your protection strategy more difficult for hackers to crack.

---

**Note:** Unused cells are those cells that appear as “unassigned” in MemView.

---

6. Click **Program Key**.

The key is programmed with the protection strategy you defined.

7. If the programming was successful, disconnect the key from the port.

---

**Note:** To determine if a programming failure is due to a software error or a hardware error, try programming another key with the same strategy. If the programming is successful, the previous error was hardware-related. If you try programming many keys, and all of them fail programming, the error is software-related. Refer to Appendix B, “Troubleshooting,” on page 331 for help, or contact Rainbow Technologies Technical Support for additional assistance.

---

8. Repeat steps 3 through 7 until all required keys have been programmed.

## Viewing Programming Statistics

After each key is programmed, the Summary section is updated accordingly:

- **Pass:** The number of keys that have been successfully programmed.
- **Fail:** The number of keys that have not been successfully programmed.
- **Total:** The total number of keys you have programmed during this session, whether they passed or failed.
- **Elapsed Time:** The amount of time it took to program the last key.

These statistics are not project-specific—even if you program multiple keys with multiple projects (protection strategies), the statistics do not reset until you exit the SSP Toolkit and start it again with a new session.

To reset the statistics without closing and restarting the SSP Toolkit, click **Reset Statistics**.

## **Verifying the Key Was Programmed Correctly**

We recommend periodically checking your product keys to be sure they are being programmed correctly.

To do so, attach a randomly selected key to a workstation running the SSP Toolkit, then open the MemView section of the API Explorer (in the Overview or Implementation stage) to graphically view each cell's access code and value. See “Viewing Memory Cells” on page 127 for more information about using MemView.

---

## Setting Up to Program Distributor Keys

Distributor keys are used to meter the activation or update of your application(s) by your distributors. You need to program one distributor key per distributor, per product line. When you program the key, you assign the initial activation counter value (the number of activations or updates the distributor has paid for) for each application being sold by your distributor.

---

**Note:** *For more information about using the Distributed activation type, see page 60.*

---

Before you begin programming distributor keys, make sure you have an adequate stock of keys with your assigned developer ID. The keys you will be programming as distributor keys *must* have the same developer ID as the key used while you designed your protection strategy.

If you need additional keys, please contact your Rainbow Technologies sales representative or distributor.

### Selecting the Appropriate Keys

Distributor keys can be stand-alone or network keys; the type of key used to create a distributor key is unimportant, as long as the developer ID is the same as the key used to create your protection strategy.

### Connecting the Keys

As you program multiple keys, you will be taking SentinelSuperPro keys on and off your workstation frequently. As a result, you will want to protect connectors from being damaged, while at the same time making it convenient to connect and disconnect keys.

---

**Note:** *You cannot cascade keys while they are being programmed. Only one key can be attached at a time during programming.*

---

To solve both issues, we recommend attaching a shielded cable with appropriate connectors for the key you are programming that reaches from the

port to a work surface, preventing you from having to reach or bend over to connect or disconnect keys.

Do not remove a key while it is being programmed, as a write failure will occur. A key removed during programming can be reprogrammed.

### ***Using Cables with the SentinelSuperPro Hardware Key***

Due to the large variety of cables currently on the market, Rainbow Technologies does not recommend a specific brand or type of cable for use with the SentinelSuperPro key, nor do we guarantee that all cables will be compatible with the key.

However, we do recommend the following:

- Cables should not be longer than 6 feet in length.
- Cables should be shielded.
- Do not use ribbon cables.
- Cables must be straight-through; that is, they must have all pin signals wired through to the connectors on either end of the cable.

Please be aware that cable connectors may only be used for a specific number of times, perhaps for as few as 100 connections. Contact the manufacturer of your cable/connector for specific information on how many insertions the cable/connector is rated for. Based on that specification, change the cable/connector on a regular basis, as needed.

Additionally, since the insertion life of the connector on the computer you are using for programming is also limited, you may want to consider using Rainbow Technologies' key programming services as a solution for reliable, high-volume key programming.



---

## Programming a Distributor Key

To program a distributor key:

1. In the SSP Toolkit, open the SentinelSuperPro project containing the protection strategy for the application you are programming a key for.
2. Navigate to the **Make Keys** stage.

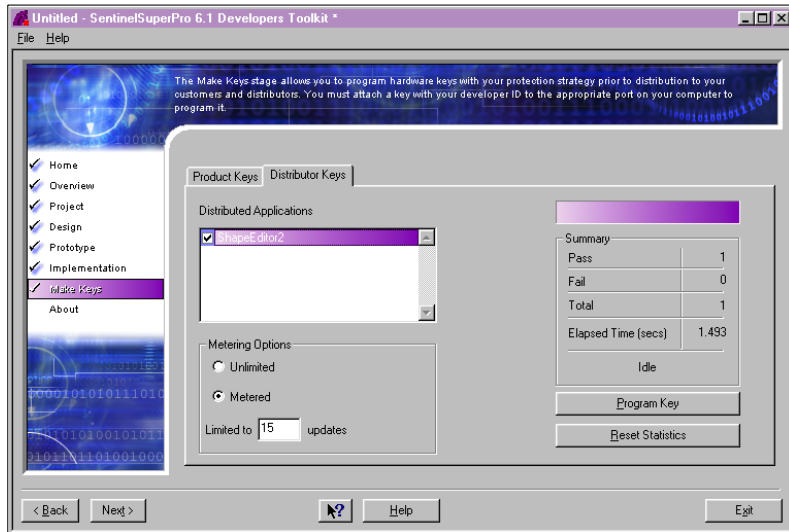
---

**Note:** *If you did not complete the Prototype stage, the Make Keys stage is unavailable and you are not ready to program product keys. Refer to Chapter 9, “Implementing Your Strategy,” on page 187 for more information on completing the Prototype stage.*

---

3. Click the **Distributor Keys** tab.

A list of the applications you applied integrated or automatic protection to, *and* assigned the Distributed activation type to, appears. You can program distributor keys *only* for those applications using the Distributed activation type.



### Make Keys Stage – Distributor Keys Tab

4. Connect the key you want to program to the appropriate port on your workstation.
5. In the **Distributed Applications** list, select the check box for the application you want to assign metering options for.

---

**Tip:** A check mark must appear in the box for the application to be selected; if the check mark does not appear, the application will not be programmed onto the key. Be sure to select the check box and not just the application name.

---

6. Under **Metering Options**, select one of the following:
  - **Unlimited:** To allow the distributor to activate or update as many of your products as they like.
  - **Limited:** To pre-define the number of applications the distributor can activate or update. Enter a number in the corresponding field.

7. Repeat steps 5 and 6 for each application you want the distributor to be able to activate and update.

You can program a distributor key to activate or update multiple applications. When you select the check box for the next application, the metering option changes to the default value of zero, or to the value you previously selected for the application.

8. Click **Program Key**.

The key is programmed with the protection strategy you defined.

9. If the programming was successful, disconnect the key from the port.

---

**Note:** *To determine if a programming failure is due to a software error or a hardware error, try programming another key with the same strategy. If the programming is successful, the previous error was hardware-related. If you try programming many keys, and all of them fail programming, the error is software-related. Refer to Appendix B, “Troubleshooting,” on page 331 for help, or contact Rainbow Technologies Technical Support for additional assistance.*

---



# Chapter 12

---

## Shipping Your Application

When your application is complete, and your product keys are programmed, you are ready to ship your protected application to your distributors and/or customers.

What you ship depends on who you are sending the application to. Customers usually only need the application, the key and the Sentinel driver.

However, when you ship your application to distributors, there are additional items you must ship along with your application and the product key, such as the stand-alone utilities used for field activation or key programming, and a distributor key for activating distributed applications.

This chapter provides lists of recommended items to send to distributors and customers; you can modify these lists as appropriate for use with your application.

To protect your product keys against damage during shipping, this chapter also provides guidelines for handling and packaging your keys.

This chapter covers the following topics:

- What to send to your customers
- What to send to your distributors
- Packaging and handling guidelines for product keys

---

## What to Send to Your Customers

When you ship your protected application to your customers, you must provide the following:

- The application executable and associated data files.
- The SentinelSuperPro Server (required for both stand-alone and network applications). See “Installing the SentinelSuperPro Server” on page 230.
- The SentinelSuperPro Monitoring Tool executable file—*monitor.exe*—and Help file—*SSP 6\_1 Monitoring Tool.chm*— if your application is a network application.
- The *SentinelSuperPro System Administrator’s Guide* in PDF format (*SentinelSP6.1 Sys Admin Guide.pdf*).
- The Sentinel Client Activator (recommended) or Field Exchange Utility (*fieldexutil.exe*) if your application is a demo or is programmed to allow field activation.

---

**Note:** *If you choose to use the Sentinel Client Activator, you also need to ship your customers the Client Activator configuration file (activator.rac) and installation tool (ainst.exe). See the Client Activator documentation for more information.*

---

- The following file: *usafe32.dll*.
- The Field Exchange Utility Help file—*Fieldexchutil.chm* (only if you are shipping the Field Exchange Utility, *fieldexutil.exe*).
- If you are shipping the Monitoring Tool or the Field Exchange Utility, the following files to provide support for HTML Help: *hhupd.exe* and *hhactivex.dll* (see page 231 for more information).
- Instructions for using the Client Activator or Field Exchange Utility (*fieldexutil.exe*) for field upgrades.
- One or more SentinelSuperPro keys programmed with the license limits and values expected by your application.

- Instructions for attaching the key to the computer or network server.
- The Sentinel system driver—*sentinel.sys* for Windows NT/2000 or *sentinel.vxd* for Windows 95/98/ME—version PD-5.39 or later.
- If you are protecting data files for a Windows 95/98/ME application, you also need to ship the Sentinel data protection driver. See “Installing the Sentinel Data Protection Driver” on page 236.
- If you wrote a separate utility for entering activation passwords, you also need to ship that utility and its associated files.

### Installing the Sentinel System Driver

The Sentinel system driver is a required component for all users of your protected application, the SentinelSuperPro Server and any Sentinel utilities, such as the Field Exchange Utility or the Make Keys Utility. To simplify the driver installation process for your customers and distributors, you may choose to incorporate the driver installation in your application’s installation routines.

The Sentinel system driver installation media is located on the SentinelSuperPro installation CD at (assuming E: is the drive letter of your CD-ROM drive): **E:\Sentinel System Driver**.

The Sentinel driver installation uses the Windows Installer that was released with Windows 2000, which supports installation on all Windows platforms back to Windows 95. Rainbow Technologies’ has also included merge modules to seamlessly integrate driver installation into your own product installation; these merge modules can be found on the SentinelSuperPro Installation CD at: **E:\Sentinel System Driver\Win CD\Merge Modules**.

If you are installing the Sentinel System Driver on DOS, Windows 3.x, OS/2 or any non-Intel Windows NT computer, you must use the old installer program, which is found in the **\Legacy** directory of the driver media. This installer is also provided for developers who are not yet ready to migrate to the Windows 2000 Windows Installer program.

Complete instructions for installing the driver, or incorporating the driver installation into your own setup routines, are provided in the *Sentinel Sys-*

*tem Driver Installation Developer's Guide*, an online HTML document that is also included on the SentinelSuperPro Installation CD. This document, which can be viewed with any Internet browser, can be found at: **E:\Sentinel SystemDriver\Win CD\SentinelDriverInstall\_Start.htm**.

### Installing the SentinelSuperPro Server

Every user of your application must also install the SentinelSuperPro server on their workstation, whether your application is a stand-alone application or a network application. Additionally, anyone who will be using a SentinelSuperPro utility, such as distributors using the License Generator Utility, or manufacturing personnel using the Make Keys Utility, also need to install the SentinelSuperPro server.

---

**Note:** *The Sentinel System Driver must already be installed prior to installing the SentinelSuperPro Server. If you will be including both the server and the driver in the same installation routine, be sure the driver is installed before the server.*

---

### Including the Server in Your Own Installation Program

To simplify the server installation process for your customers, you may choose to include the server installation as a part of your application's setup routines, so that the server is installed automatically.

---

**Note:** *You can also create your own installer program that includes just the SentinelSuperPro Server and the Sentinel driver, and have your users run it separately from your application installer. How you install the server and driver is up to you, as long as they do get installed.*

---

For your convenience, merge modules for use with the Windows Installer have been provided on the SentinelSuperPro Installation CD. These modules can be found at: **E:\SentinelSuperProServer\Merge Modules**. The following procedure provides instructions for using the merge modules in your own installer.

If you choose to not use Windows Installer, another alternative is to add the files manually to your own installer and then use the *loadserv.exe* utility to



add the Windows NT service, and manually add shortcuts for the Windows 9x (95/98/ME) server. Use the information on page 233 to help you manually add the executable files into your installation program.

## Using Merge Modules with Windows Installer 1.1 or Later

Three merge modules are provided to allow you to control what gets installed on the user's system. The three files are:

- **SuperProNetServers.msm:** This module contains the actual server programs for Windows 9x and NT. It determines which server to install depending on the operating system it is being installed on.

On Windows NT, it installs the service for you and starts it. On Windows 9x, it adds a shortcut in the startup group so that it starts automatically on the next boot.

The server component requires the Sentinel driver v.5.39 or higher to already be installed on the user's system. Merge modules for the driver are also provided—these can be placed into your installation package so that both the server and driver are installed at the same time. Refer to the *Sentinel System Driver Installation Developer's Guide*, an online HTML document included on the SentinelSuperPro Installation CD, for more information.

- **SuperProMonitor.msm:** This module contains the Sentinel SuperPro Monitoring Tool, which is used to monitor SentinelSuperPro key usage across the network. It is supplied in a separate merge module so that it can be optionally installed on the user's system. Typically, the Monitoring Tool only needs to be installed with the server when your application is being run as a network application.
- **MicrosoftHTMLHelp.msm:** This module contains the files necessary to be able to view the HTML Help files included with the SentinelSuperPro utilities and the Monitoring Tool. To provide support for Microsoft HTML Help functionality on end-users' systems, Microsoft HTML Help system files need to be installed and registered on their computers.

These files (*hhupd.exe* and *hhactivex.dll*) are installed permanently; they are not removed by uninstalling SentinelSuperPro. Also, because these files are standard with Windows 2000, you do not need to include them in your installer if all your users will be running your application on Windows 2000.

---

**Note:** *To view HTML Help, users also must have Microsoft Internet Explorer 4.0 or later installed on their systems. Internet Explorer is NOT part of this merge module, and may need to be installed separately. Refer to [www.microsoft.com](http://www.microsoft.com) for more information.*

---

The merge modules can be added to any Microsoft Windows Installer package. There are many third-party applications that can be used to design a Windows Installer package. An example would be InstallShield® for Windows Installer. As with other applications, it assists you in building the actual installation media and simplifies configuration. For more information about using and developing a Windows Installer package, visit Microsoft's MSDN Web site at <http://www.microsoft.com>.

Once you add the merge module to your installation package, you can associate it with a particular installation feature. This allows the user to have some control over whether this feature should be installed. Alternatively, you can make it a required item so that the user must install it with your application.

The merge module itself contains the necessary files, registry entries and shortcuts to install the server. It will determine what needs to be installed based on the operating system it is being installed on. This frees you from having to understand exactly how to install the server, yet you can install the server from your own installer, giving you full control of the installation.

There are no special custom actions or properties that have to be used with these merge modules. The only supported properties are the standard properties that are used by any Windows Installer package. Please refer to the Microsoft Windows Installer documentation for more information.

## Requiring a Reboot During Server Installation

The server has no special requirements that would require a reboot to start up the server. However, since it does require the Sentinel System Driver 5.39 or higher to be installed to operate correctly, the driver reboot requirements may affect whether the server can be started.

Installing the Sentinel System Driver on Windows NT never requires a reboot; therefore the Windows NT SuperPro Server is always started as part of the installation process. However, Windows 9x sometimes requires a reboot to start the Sentinel System Driver and in those cases, the server cannot be started during installation. For this reason, no attempt is made to start the Windows 9x server as part of the installation. If your installer determines that a reboot is not required for the Sentinel System Driver, it is safe to run the Windows 9x server executable at the end of the installation.

For more information about when the system must be rebooted during a Windows 9x Sentinel System Driver installation, refer to the online *Sentinel System Driver Installation Developer's Guide*.

## ***Installing the Server with the Executable Files***

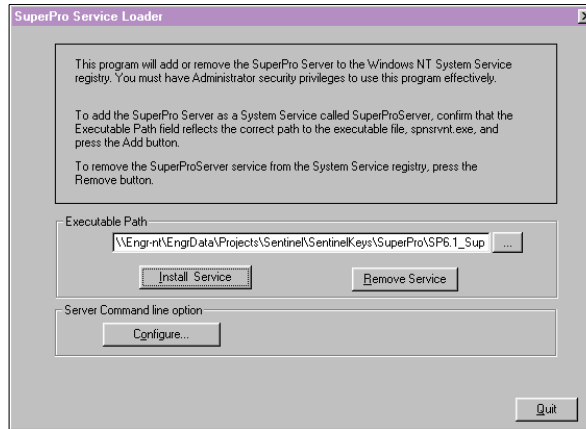
For your distributors and manufacturing employees, you can simply provide the server executable file and ask them to run the file on their workstation to install the server. The executable files can be found on the SentinelSuperPro Installation CD at: **E:\SentinelSuperProServer\Win9x** or **E:\SentinelSuperProServer\WinNT**.

Windows 95, 98 or ME users should use the *spnsrv9x.exe* and *loadserv.exe* files. Windows NT or 2000 users should use the *spnsrvnt.exe* and *loadserv.exe* files. Instructions for installing both server types are provided below.

To install the Windows 9x server:

1. Verify that the Sentinel system driver has already been installed on your system.
2. Copy the *spnsrv9x.exe* and *loadserv.exe* files to any location on your local hard drive.

3. Double-click on *loadserv.exe* to run the file. The following screen appears:



### Server Installation Screen

4. Ignore the **Executable File** field.

The path in this field is used for installing a server on Windows NT or 2000 only.

5. Click **Configure** to set the path for the server log file.

The server checks for the log file location when it starts up; if no path for a log file is set, logging will not occur.

6. Click **Exit**.

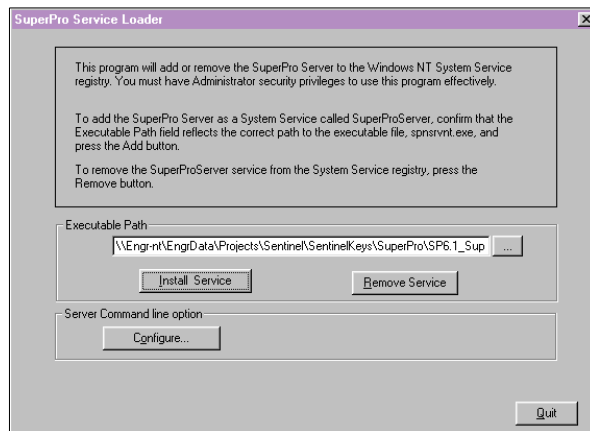
7. Right-click on the *spnsrv9x.exe* file. A shortcut menu appears.

8. From the shortcut menu, select **Create Shortcut**. A shortcut to the *spnsrv9x.exe* file appears in the same directory.

9. Copy the shortcut and move it to the following directory:  
**Win32\Start Menu\Programs\Startup**. This will start the server automatically whenever your system is booted up.

To install the server on Windows NT or 2000:

1. Verify that the Sentinel system driver has already been installed on your system.
2. Copy the *spnsrvnt.exe* and *loadserv.exe* files to any location on your local hard drive.
3. Double-click on *loadserv.exe* to run the file. The following screen appears:



### Server Installation Screen

4. Confirm that the path in the **Executable File** field is the correct path for the location of the *spnsrvnt.exe* file.
5. Click **Configure** to set the path for the server log file.
6. Click **Install Service** to install the server. The server is an NT service that will start automatically whenever your system is booted up.

## Installing the Sentinel Data Protection Driver

The Sentinel data protection driver implements a file system which intercepts all file I/O at the operating system level. **Without this driver loaded at the end user's Windows 95/98/ME workstation, the protected application will not be able to decrypt files.**

If you protect data files for Windows 95/98/ME applications, your customers need to receive and install the data protection driver, as well as your application and data files.

For your convenience, Rainbow Technologies has included an installation program for the Sentinel data protection driver on the SentinelSuperPro installation CD, located at (assuming E: is the drive letter of your CD-ROM drive): **E:\Data Protection Driver**.

Also, so you can modify this installation program for your own installation needs, we have provided the C source code that installs this program. Please refer to the source code file for information on modifying it.

---

**Note:** *The data protection driver must only be run on Windows 95/98/ME; it is the responsibility of the calling program to check for the type of operating system being used. Remind customers that their computer must be rebooted after installing the data protection driver in order to load the driver.*

---

The following files comprise the Sentinel data protection driver:

- **instdrv.exe:** An installation program that installs the Sentinel system driver in the correct location on your customer's workstation.
- **instdrv.c:** The C source code file that installs the program and creates the registry keys. It may be modified to suit your needs.
- **sentdata.vxd:** The Sentinel data protection driver needed to perform transparent data file encryption/decryption in 32-bit applications running in Windows 95/98/ME.

## ***Calling the Installation Program***

Your installation program may call the data protection driver with the following command line options:

- **/P** Specifies the source path for *sentdata.vxd*. If not included, the installer looks for the .VXD file in the directory where the driver installer resides.
- **/U** Uninstalls the driver.

The installation program returns a zero if it was successful; otherwise, it returns the error code of the last Win32 API call that had an error (if applicable). If the unsuccessful API call does not return an error, the installer returns a -1.

---

## What to Send to Your Distributors

If your company uses distributors to sell your products, you need to decide:

- If distributors can provide license codes for field activation.
- If distributors will be activating or updating your application using a distributor key, and if so, how many activations you will limit them to.

Your decisions about these items will determine what you ship to your distributors.

### Customer Items

These are those items you need to ship to your distributor so they can send them to customers who purchase your protected application.

- The application executable and associated data files.
- The SentinelSuperPro Server (required for both stand-alone and network applications). See “Installing the SentinelSuperPro Server” on page 230.
- The SentinelSuperPro Monitoring Tool executable file—*monitor.exe*—and Help file—*SSP 6\_1 Monitoring Tool.chm*— if your application is a network application.
- The *SentinelSuperPro System Administrator’s Guide* in PDF format (*SentinelSP6.1 Sys Admin Guide.pdf*).
- The Sentinel Client Activator (recommended) or Field Exchange Utility if your application is a demo or is programmed to allow field activation.

---

**Note:** If you choose to use the Sentinel Client Activator, you also need to ship your customers the Client Activator configuration file (*activator.rac*) and installation tool (*ainst.exe*). See the Client Activator documentation for more information.

---

- The following file: *usafe32.dll*.



- The Field Exchange Utility Help file—*Fieldexchutil.chm* (only if you are shipping the Field Exchange Utility, *fieldexutil.exe*).
- If you are shipping the Monitoring Tool or the Field Exchange Utility, the following files to provide support for HTML Help: *hhupd.exe* and *hhactivex.dll* (see page 231 for more information).
- Instructions for using the Client Activator or Field Exchange Utility for field upgrades.
- One or more SentinelSuperPro keys programmed with the license limits and values expected by your application.
- Instructions for attaching the key to the computer or network server.
- The Sentinel system driver—*sentinel.sys* for Win NT/2000 or *sentinel.vxd* for Win 95/98/ME—version PD-5.39 or later.
- If you are protecting data files for a Windows 95/98/ME application, you also need to ship the Sentinel data protection driver. See “Installing the Sentinel Data Protection Driver” on page 236.

## Distributor-Only Items

The items in this section are shipped to distributors for their use only, if you have decided to allow them to activate applications or generate license codes for field upgrades.

- A distributor key, if you want to manage the number of activations or updates your distributor can perform.
- The License Generator Utility (*LicenseGenUtil.exe*) and instructions for using it.
- The following files: *lang\_enu.dll*, *sp\_gXX.dll*<sup>1</sup>, *spcommon.dll*, *dsafedll.dll* and *dsafe32.dll*.
- The SentinelSuperPro distributor’s project file (.DST) with the protection strategy for the application you are shipping.

---

1. The *sp\_gXX.dll* file can be any of the following, depending on the color depth the display driver is configured to: *sp\_g24.dll*, *sp\_g08.dll* or *sp\_g04.dll*.

### ***Documentation***

You can copy the appropriate sections from Chapter 14, “Using the Stand-alone Utilities,” on page 259 to send to distributors using the License Generator Utility, or to customers using the Field Exchange Utility. You can also use the information in Chapter 14 to create your own documentation for these utilities.

---

## Packaging and Handling Guidelines for Keys

The following information includes guidelines for shipping SentinelSuperPro keys with your protected application. Following these guidelines should help ensure that the keys you ship reach your customers safely.

In general, the shipping and handling procedures suggested for the SentinelSuperPro keys conform to the industry standards for handling electronic printed circuit boards. Your company may already have appropriate work surfaces and procedures in place.

Use the following steps to prepare the work areas used to receive, inspect, stock, program and package SentinelSuperPro keys:

- Install a electrostatic-dissipating mat for a work surface. Make sure the mat is properly grounded.
- Ensure all operators wear grounding wrist or ankle straps.
- When storing SentinelSuperPro keys for inventory, use plastic tubs designed to dissipate electrostatic charges.
- Use packaging materials designed to avoid electrostatic charge during shipment. Plastic that does not generate static (called *cold plastic*) is typically pink in color. You may also choose to use *conductive plastic*, which is designed to drain off static.

---

---

**Warning!** *Electrostatic charges may damage the SentinelSuperPro keys. Work surface mats and wrist straps are strongly recommended.*

---

---



# Chapter 13

---

## Activating and Updating Keys

SentinelSuperPro's **Secure, Authenticated Field Exchange** (SAFE, also known as *field activation*) protection system provides you with a secure method of remotely updating a SentinelSuperPro hardware key's memory after the key is sent to your user or distributor.

Field activation allows you to increase demo limits, upgrade demo applications to fully licensed versions, and provide access to additional modules or features, without having to ship a new key to the customer or visit the customer's site. It also allows you to update distributor keys to add activations in the field.

In the field, your customers generate a locking code that they send to you (or your distributor). You then input the locking code in the Field Activation section of the Implementation stage (or the License Generator Utility) to generate a license code that you return to the customer. The license code updates the key and activates the customer's application appropriately.

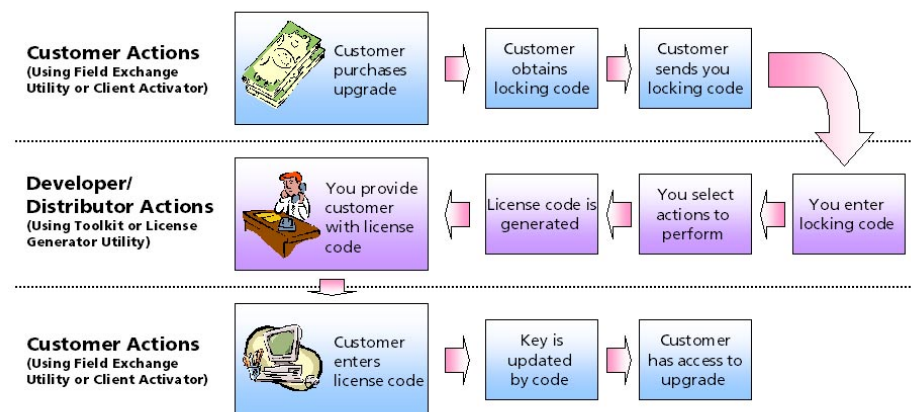
This chapter covers the following topics:

- How keys are activated or updated
- Using the Sentinel Client Activator
- Receiving customer locking codes
- Generating license codes
- Updating distributor keys

## How Product Keys are Activated or Updated

SentinelSuperPro keys in the field are updated remotely as a result of information exchanged between you and your customer. All exchanged information is encrypted and secure, and cannot be used to update any other SentinelSuperPro key.

The key update process is as follows:



### Customer and Developer/Distributor Roles in Key Activating or Updating

1. Customer purchases additional services, such as a higher license limit, or added features.
2. Customer uses the Sentinel Client Activator or the SentinelSuperPro Field Exchange Utility to generate a *locking code*.
3. Customer sends the locking code to you (or your distributor, if you authorized your distributor to perform field activation) via telephone, fax, e-mail or the Internet.
4. You (or your distributor) enter the locking code in the Field Activation section of the SSP Toolkit Implementation stage or in the SentinelSu-

perPro License Generator Utility. The key information is extracted from the locking code.

5. You select the actions you want to perform on the key in the field.

---

**Note:** *Available activation and update actions, and their corresponding commands, were defined when the protection strategy for the application was being designed. See Chapter 10, “Defining Field Activation Actions,” on page 201 for more information.*

---

6. SentinelSuperPro generates a *license code*—specific to the customer’s key—based on the locking code you entered and the actions you selected.
7. You send the license code back to your customer.
8. Customer enters the license code in the Client Activator or Field Exchange Utility.

The license code tells the Field Exchange Utility or Client Activator how to reprogram the key, including how to activate the application.

9. The key is updated and the customer has access to the services he purchased.

### What Is a Locking Code?

The locking code for a key includes information about how the key is currently programmed, including the key’s serial number and developer ID. You must have a customer-generated locking code to create a license code.

Locking codes are unique for each key.

---

**Note:** *In previous versions of SentinelSuperPro, the locking code was known as the Key ID string.*

---

## What Is a License Code?

The license code for a key describes the actions to be performed on a key in the field. It determines how the application will be activated or updated; for example, what new features the customer will have access to, or the number of additional licenses that will be added.

The license code is generated by SentinelSuperPro based on the locking code provided by the customer and the actions you select. When the customer enters the license code in the Client Activator or Field Exchange Utility, a script is automatically run that performs the selected actions on the key.

License codes are unique to the key the locking code was generated from.

---

**Note:** *In previous versions of SentinelSuperPro, the license code was known as the Update Key string.*

---



---

## How Distributors Activate an Application

Distributors activate an application in much the same way that you would update a key in the field. They use the License Generator Utility, together with the .DST file you provide them, to create license codes that activate or update your protected application. Distributors *must* have the SentinelSuperPro server running and have their distributor key connected to their system while using the License Generator Utility to generate license codes.

Distributor responsibilities are dependent on how much work you want to off load to your distributors. SentinelSuperPro allows for many distribution models to be used; any of the following models could be implemented:

- The distributor activates product keys prior to sending them to your customers.
- The distributor sends non-activated product keys to your customers, and activates the key through field activation after the customer receives it.
- The distributor is responsible only for providing application updates through the field activation process.

Distributors activate product keys through the following process:

1. Customer uses the Sentinel Client Activator or the SentinelSuperPro Field Exchange Utility to generate a *locking code*.
2. Customer sends the locking code to the distributor via telephone, fax, e-mail or the Internet.
3. With his distributor key connected and the SentinelSuperPro server running, the distributor opens the .DST file you provided in the SentinelSuperPro License Generator Utility.
4. The distributor enters the locking code in the License Generator Utility. The key information is extracted from the locking code.
5. The distributor selects the actions to perform on the key in the field.

6. SentinelSuperPro generates a *license code*—specific to the customer's key—based on the locking code entered and the actions selected.
7. The activation counter on the distributor's key is decremented by one after the license code is generated.
8. The distributor sends the license code back to your customer.
9. Customer enters the license code in the Client Activator or Field Exchange Utility.
10. The key is updated and the customer has access to the application.

---

## Using the Client Activator

The Sentinel Client Activator is an automated license installation utility that is used to create a product-specific activation script for your protected application.

The Client Activator is Rainbow Technologies' recommended means of field activation for SentinelSuperPro protected applications, due to its user-friendly interface. The Client Activator also allows your customers to easily and quickly activate your product via a Web site, if you desire. Additionally, if you are going to use SentinelExpress with your SentinelSuperPro-protected application for field activation, you must use the Client Activator.

Included in the Client Activator is an Auto Activation Wizard that builds and configures a Client Activator for a specific protected application. The Wizard collects product and publisher information that is used by the Client Activator to process a license activation request.

The Wizard allows you to choose how the product activation will be presented to your customer, and defines the methods your customer can use to activate the product (keyboard/file, telephone, fax/mail, Internet and drop-in hardware).

To build your product-specific Client Activator, you simply define your product and activation method(s). The Wizard builds the Client Activator, which you ship with your protected application. When your customer installs the application, he or she has the option of clicking the **Try** or **Buy** button.

The **Try** button allows your customer to use the product for a specified time limit or pre-determined number of executions. The **Buy** button prompts your customer for the necessary information and completes the activation.

---

**Note:** *The Client Activator is a stand-alone product that is included with the SSP Toolkit. Client Activator must be installed separately; refer to the Client Activator documentation for more information.*

---

## Client Activator Customer Requirements

Your customers must be running one of the following operating systems in order to use the Client Activator:

- Windows 98/ME
- Windows NT 4.0
- Windows 2000

Additionally, if you are going to allow product activations and upgrades over the Internet, your customers must use one of the following Internet browsers:

- Internet Explorer 4.01 or 5.0
- Netscape Navigator 4.6 or 4.7

## Steps for Deploying the Client Activator

If you decide to use the Client Activator to activate your application in the field, you need to complete the following steps:

- Protect your application with the SSP Toolkit.
- Use the Activation Wizard to tell the Client Activator how to activate your application. You'll need to provide the Client Activator with the *usafe32.dll* that was created during the Prototype stage (see Chapter 9, "Implementing Your Strategy," on page 187 for more information on the Prototype stage).
- Include the *.rac* and *ainst.exe* files in the application's install program.
- Modify the install program to run *ainst.exe* during installation.
- Package and ship the protected application as described in Chapter 12, "Shipping Your Application," on page 227 of this guide.

For more information about using the Client Activator, including the complete list of files you need to ship to your customer, please refer to the Client Activator documentation.

## ***Where to Install Client Activator***

In order to update keys, the Client Activator must be installed on the same system where the SentinelSuperPro key has been connected.

- If your application is a stand-alone application, the Client Activator should be installed on each client system.
- If your application is a network application, the Client Activator should be installed only on the server where the SentinelSuperPro key is located; it does not need to be installed on the client systems.

Be sure to modify your install program appropriately.

## **If You Don't Use the Client Activator**

If you decide to not use the Client Activator for field activation and upgrades, your customers will need to use the SentinelSuperPro Field Exchange Utility to obtain locking codes. This utility is also included on your SentinelSuperPro installation CD. See Chapter 14, "Using the Stand-alone Utilities," on page 259 for more information.

---

## Updating Product Keys in the Field

Field activation requires both you and your customer to exchange information about the key. Your customer is responsible for generating and sending the locking code to you. You are responsible for generating and sending the license code to the customer.

This section provides instructions for the developer's or distributor's role in field activation. For more information and detailed instructions on the customer's role, please refer to Chapter 14, "Using the Stand-alone Utilities," on page 259.

### Receiving the Locking Code from Your Customer

Before you can generate a license code, you **must** receive a locking code from your customer for the key protecting the application they want to update. If the customer has multiple copies of your application, each of which use a different key, your customer must send you a unique locking code for *each* key.

There are several ways that a customer can provide you with the locking code; use the method that works best for both you and your customer:

- Telephone
- E-mail
- Fax
- Internet

Also, your customer can save the code to a file. This file has an default name of *LockingCode.loc*, and can be loaded directly into the License Code Generator.

### Generating a License Code

Once you have received the customer's locking code, you are ready to generate a corresponding license code.

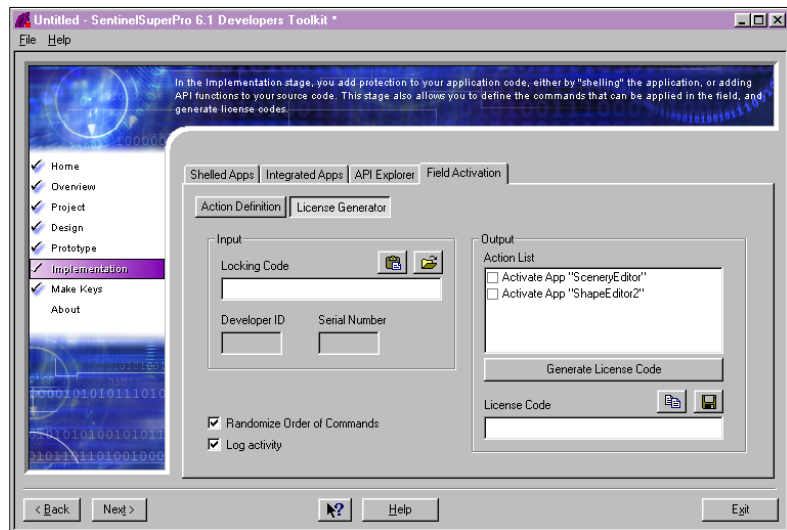
---

**Note:** The instructions in this section assume you are using the Field Activation section of the Implementation stage to generate the locking code. If you are using the License Generator Utility, refer to Chapter 14, “Using the Stand-alone Utilities,” on page 259 for more information.

---



To generate a license code:

1. If you are using a distributor key, connect the key to the appropriate port.
2. In the SSP Toolkit, open the **Implementation** stage.
3. Click **Field Activation**.
4. Click **License Generator**. The License Code Generator appears.



### License Code Generator

5. Under **Input**, in the **Locking Code** field, enter the locking code provided by your customer.

- If you copied the locking code to the clipboard, click the paste button  to paste the code in the field.
- If the locking code was saved to a file (.LOC), click the open button  to locate and open the file. The code is entered in the field automatically.

The developer ID and serial number of the key the code was generated from appears in the corresponding fields.

6. Under **Output**, in the **Action** list, select the action(s) you want to perform on the customer's key.

You can select as many actions as necessary. To remove an action, clear the corresponding check box.

---

**Note:** Only selected actions (identified with a check mark) are added to the license code's script. Remember, actions determine which application features will be activated or upgraded. For more information about actions, see Chapter 10, "Defining Field Activation Actions," on page 201.

---

7. If you don't want to randomize the order in which commands are applied to the key in the field, clear the **Randomize Order of Commands** check box.

While randomizing the order commands are applied in can make it harder for hackers to trace what you are doing in code, there are certain situations in which you should *not* randomize commands.

Assume you have two commands that operate on the same cell. One command must be executed before the other because the second command expects the cell to be in a state set by the first command. In this case, you should *not* randomize commands.

Randomizing commands can produce a different license code for a given action set, even if the locking code is the same. Non-randomized commands produce the same license code every time for any given locking code and action set.





8. Select the **Log Activity** check box to create a log file (*LicenseGen.log*) containing all the license codes you generate.

9. Click **Generate License Code**.

SentinelSuperPro creates a unique license code based on the locking code and the actions you selected in step 6. The license code appears in the License Code field.

10. If necessary, do one of the following:

- To copy the license code to the clipboard, click the copy button  .
- To save the license code to a file, click the save button  and select a file name and location for the license code file.

11. Go to the next section to send the license code to your customer.

## Sending the License Code to Your Customer

Once the license code has been generated, you need to send it to your customer. You have several options for how to send it; use the method that works best for both you and your customer:

- Telephone
- Fax
- E-mail

When your customer receives the license code, they must enter it in the Client Activator or the Field Exchange Utility to update or activate their key and receive their updates.

---

**Note:** *If you selected the one-time update option, your customer will only be able to apply the license code one time. If they attempt to apply the license code more than once, no additional updates or counter increments will occur.*

*However, if you did **not** select the one-time update option, your customer can apply the license code as many times as they like, possibly incrementing counters more than you intended. For more information about using the one-time update option, see “Enabling the One-Time Update Option for License Codes” on page 115.*

---

For information on how your customer enters the license code using the Field Exchange Utility, refer to Chapter 14, “Using the Stand-alone Utilities,” on page 259.

For information about using the Client Activator to enter the license code, please refer to the Client Activator documentation.

---

**Tip:** *Remember, you should provide your customers with instructions for generating the locking code and entering the license code. See “What to Send to Your Customers” on page 228 for more information.*

---

---

## Updating Distributor Keys in the Field

When the activation counter on a distributor key reaches zero, the distributor will no longer be able to activate your application, or perform updates to keys in the field. You can increment the activation counter on a distributor key (and charge for doing so) in the same way that you update product keys in the field.

To update a distributor key in the field:

1. Ask your distributor to run the Field Exchange Utility, while his distributor key is connected to his system, to generate a *locking code*.
2. Tell your distributor to send the locking code to you via telephone, fax, e-mail or the Internet.
3. Enter the locking code in the Field Activation section of the SSP Toolkit Implementation stage. The key information is extracted from the locking code.
4. Select the **Increment Distributor Counter** action.
5. Click **Generate License Code**. SentinelSuperPro generates a *license code*—specific to the distributor's key—based on the locking code you entered and the action you selected.
6. Send the license code back to the distributor.
7. Distributor enters the license code in the Field Exchange Utility.
8. The key is updated and the distributor has access to additional activation or update licenses.



# Chapter 14

---

## Using the Stand-alone Utilities

SentinelSuperPro comes with a set of three stand-alone utilities that allow you to give other people—such as manufacturing department employees or distributors—the ability to perform selected SentinelSuperPro functions without also having access to your passwords and protection strategy information. The three utilities are as follows:

- The *Make Keys Utility* allows you to program keys for your protected application.
- The *License Generator Utility* allows you or your distributors to activate and update product keys in the field through creation of a license code.
- The *Field Exchange Utility* is used by your customers to generate locking codes and enter license codes. This utility is necessary only if you are using field activation for your application, but are not shipping the Client Activator.

---

**Tip:** For information about the stand-alone SentinelSuperPro Monitoring Tool, please refer to the SentinelSuperPro System Administrator's Guide, included in your package.

---

## Chapter 14 – Using the Stand-alone Utilities

This chapter covers the following topics:

- Verifying the SentinelSuperPro Server is running
- Using the Make Keys Utility
- Using the License Generator Utility
- Using the Field Exchange Utility

---

## Verifying the SentinelSuperPro Server Is Running

To use any of the SentinelSuperPro stand-alone utilities, the SentinelSuperPro Server must be installed and running on the same workstation the utility is being used on. If the server is not running, you will be unable to program keys, generate license codes, or generate locking codes.

Before opening any of the utilities, you should verify that the SentinelSuperPro server is installed and running. To verify the server is running:

- In Windows 98/ME, from the **Start** menu, point to **Programs > Accessories > System Tools > System Information**.

In the dialog box that appears, click on **Software Environment**. Then, under **Running Tasks**, look for *spnsrv9x.exe*. If this file is listed, the server is running.

- In Windows NT, from the **Start** menu, point to **Settings > Control Panel**, then double-click on the **Services** icon. In the dialog box that appears, the SuperPro Server should be listed with a status of **Started**.
- In Windows 2000, from the **Start** menu, point to **Settings > Control Panel > Administrative Tools**, then double-click on the **Services** icon. In the dialog box that appears, the SuperPro Server should be listed with a status of **Started**.

If the server is installed, but not running, do one of the following:

- In Windows 95/98/ME, locate and then double-click the *spnsrv9x.exe* file. The server will start automatically.
- In Windows NT, from the **Start** menu, point to **Settings > Control Panel**, then double-click on the **Services** icon. In the dialog box that appears, select **SuperPro Server**, then click **Start**.
- In Windows 2000, from the **Start** menu, point to **Settings > Control Panel > Administrative Tools**, then double-click on the **Services** icon. In the dialog box that appears, select **SuperPro Server**, then click **Start**.

## Using the Make Keys Utility

To avoid giving your manufacturing department access to your passwords—which would also give them the ability to change field activation commands or other elements in your protection strategy—we recommend providing them with the Make Keys Utility.

This utility has the exact same functionality as that found in the Make Keys stage in SentinelSuperPro. See Chapter 11, “Programming Keys,” on page 213 for more information about the Make Keys stage.

### Installing the Make Keys Utility

The Make Keys Utility is installed automatically during SentinelSuperPro setup, in the same folder where you installed the SSP Toolkit. To use the Make Keys Utility, the SentinelSuperPro Server must also be installed and running on the same workstation. To distribute this utility and the server to your manufacturing department, do one of the following:

- Use the SentinelSuperPro setup program to install only the Make Keys Utility and the SentinelSuperPro Server on the appropriate computers. The setup program is accessible from the SentinelSuperPro installation CD. See Chapter 2, “Installation,” on page 19 for more information.
- Provide them with a copy of the Make Keys Utility executable file (*MakeKeysUtil.exe*), and the following files: *spcommon.dll*, *lang\_enu.dll*, *sp\_gXX.dll*, *makedll.dll* and *makekeysutil.chm*. All files should be placed in a single directory.

Also, provide them with a copy of the appropriate server executable file and ask them to run the file on their workstation to install the server. Windows 95, 98 or ME users should use the *spnsrv9x.exe* file. Windows NT or 2000 users should use the *spnsrvnt.exe* and *loadserv.exe* files. Instructions for installing both server types are provided in “Installing the Server with the Executable Files” on page 233.



You also need to provide these users with the SentinelSuperPro project (.SPP) containing the protection strategy for the application they will be creating keys for.

You'll want to install the Make Keys Utility on computers that reside on your manufacturing floor. You are free to install a copy of this utility and the server on multiple computers within your manufacturing area, so that keys can be programmed on multiple computers at the same time.

## Opening the Make Keys Utility

To open the Make Keys Utility:

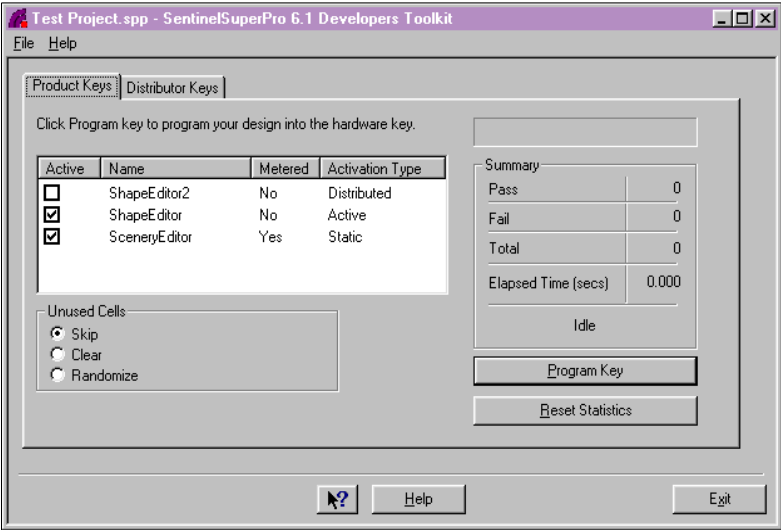
1. Verify the SentinelSuperPro Server is installed and running. See “Verifying the SentinelSuperPro Server Is Running” on page 261 for instructions.
2. Do one of the following:
  - From the **Start** menu, point to **Programs > Rainbow Technologies > SuperPro > 6.1**, then select **Make Keys Utility**.
  - Double-click the **MakeKeysUtil.exe** file icon.

---

**Note:** *If you did not use the SentinelSuperPro setup program to install the Make Keys Utility, it will not be accessible from the Start > Programs menu.*

---

A list of the applications protected in the project appears. The values under **Metered** indicates whether or not the application has a demo counter or metered options associated with it.



**Make Keys Utility – Product Keys Tab**

**Programming Product Keys**

1. From the **File** menu, select **Open**. The Open dialog box appears.
2. Browse to locate the SentinelSuperPro project file (.SPP) containing the protection strategy for the application you are programming a key for, then click **Open**.
3. Connect the key you want to program to your workstation. See “Connecting the Keys” on page 215 for more information.
4. Verify you are on the **Product Keys** tab.
5. To override the activation status of an application for this key only, select the **Active** check box for the appropriate application.

For example, you can make a demo application inactive so that it must be activated in the field.

If the **Active** check box is selected, the application will be shipped as active for this key only. If the check box is cleared, the application will be inactive upon shipment, requiring the user to enter an activation password to run the application.

---

**Note:** *Because the override is on a per key basis, if you make a change to the activation status, and then close the Make Keys Utility, when you open it again the status change is **not** saved.*

---

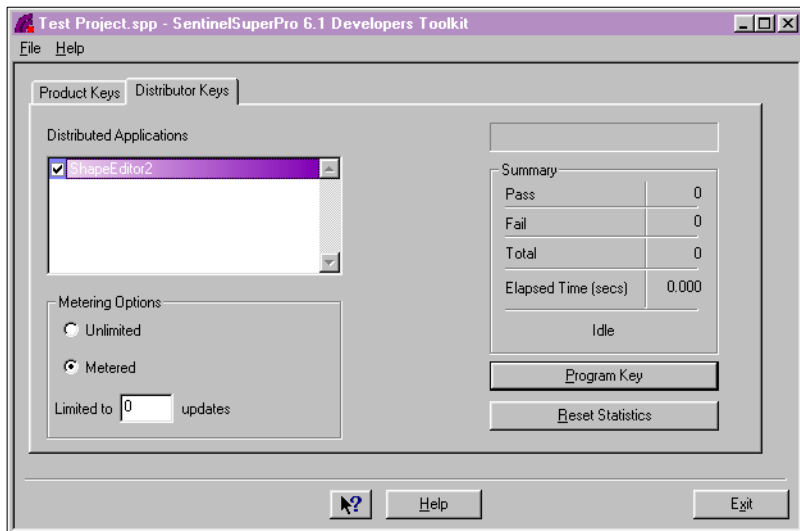
6. Under **Unused Cells**, select one of the following:
  - **Skip:** Overwrites any cells used in the protection strategy, but leaves any cells that are unallocated or not assigned values in your protection strategy. This is the default option.
  - **Clear:** Removes values and access codes/cell types from all cells not used in the protection strategy, including any previously programmed cells.
  - **Randomize:** Randomly assigns values and access codes to cells not used in the protection strategy. This is the most secure option; it makes the protection more difficult for hackers to crack.
7. Click **Program Key**. The Make Keys Utility programs the key with the protection strategy you defined.
8. If the programming was successful, disconnect the key from the port.

If the key failed the programming process, refer to Appendix B, “Troubleshooting,” on page 331 for assistance.
9. Repeat steps 3 through 8 until all keys have been programmed.

## Programming Distributor Keys

1. From the **File** menu, select **Open**. The Open dialog box appears.
2. Browse to locate the SentinelSuperPro project file (.SPP) containing the protection strategy for the application you are programming a distributor key for, then click **Open**.
3. Connect the key you want to program to the appropriate port on your workstation. See “Connecting the Keys” on page 215 for more information.
4. Click the **Distributor Keys** tab.

A list of the applications assigned the Distributed activation type appears. You can program distributor keys *only* for those applications using the Distributed activation type.



### Make Keys Utility – Distributor Keys Tab

5. In the **Distributed Applications** list, select the check box for the application you want to assign metering options for.

---

**Tip:** *A check mark must appear in the box for the application to be selected; if the check mark does not appear, the application will not be programmed onto the key. Be sure to select the check box and not just the application name.*

---

6. Under **Metering Options**, select one of the following:

- **Unlimited:** To allow the distributor to activate or update as many of your products as they like.
- **Limited:** To pre-define the number of applications the distributor can activate or update. Enter a number in the corresponding field.

7. Repeat steps 5 and 6 for each application you want the distributor to be able to activate and update.

You can program a distributor key to activate or update multiple applications. When you select the check box for the next application, the metering option changes to the default value of zero, or to the value you previously selected for the application.

8. Click **Program Key**. The Make Keys Utility programs the key with the protection strategy you defined.

9. If the programming was successful, disconnect the key from the port.

If the key failed the programming process, refer to Appendix B, “Troubleshooting,” on page 331 for assistance.

10. Repeat steps 3 through 9 until all distributor keys have been programmed.

## Viewing Programming Statistics

After each key is programmed, the Summary section is updated accordingly:

- **Pass:** The number of keys that have been successfully programmed.
- **Fail:** The number of keys that have not been successfully programmed.

---

**Note:** *To determine if a programming failure is due to a software error or a hardware error, try programming another key with the same strategy. If the programming is successful, the previous error was hardware-related. If you try programming many keys, and all of them fail programming, the error is software-related. Refer to Appendix B, “Troubleshooting,” on page 331 for help, or contact Rainbow Technologies Technical Support for additional assistance.*

---

- **Total:** The total number of keys you have programmed during this session, whether they passed or failed.
- **Elapsed Time:** The amount of time it took to program the last key.

These statistics are not project-specific—even if you program multiple keys with multiple project files (protection strategies), the statistics do not reset until you exit the Make Keys Utility and start it again with a new session.

To reset the statistics without closing and restarting the Make Keys Utility, click **Reset Statistics**.

---

## Using the License Generator Utility

If you are giving your distributors—or anyone in your organization—the ability to provide field activation and updates for your customers, we recommend giving them the License Generator Utility to do so.

This prevents those users from changing field action or command definitions, or other elements in your protection strategy, and also does not give them access to your passwords.

This utility has the exact same functionality as that found in the Field Activation > License Generator section of the Implementation stage in the SSP Toolkit. See Chapter 13, “Activating and Updating Keys,” on page 243 for more information about field activation.

### Installing the License Generator Utility

The License Generator Utility is installed automatically during SentinelSuperPro setup, in the same folder where you installed the SSP Toolkit. To use the License Generator Utility, the SentinelSuperPro Server must also be installed and running on the same workstation. To distribute this utility and the server to your distributors or other users, do one of the following:

- Use the SentinelSuperPro setup program to install only the License Generator Utility and the SentinelSuperPro Server on the appropriate computers. The setup program is accessible from the SentinelSuperPro installation CD. See Chapter 2, “Installation,” on page 19 for more information.
- Provide them with a copy of the License Generator Utility executable file (*LicenseGenUtil.exe*), and the following files: *spcommon.dll*, *lang\_enu.dll*, *sp\_gXX.dll*, *dsafedll.dll*, *dsafe32.dll*, and *licensegenutil.chm*. All files should be placed in a single directory.

Also, provide them with a copy of the appropriate server executable file and ask them to run the file on their workstation to install the server. Windows 95, 98 or ME users should use the *spnsrv9x.exe* file. Windows NT or 2000 users should use the *spnsrvnt.exe* and *loadserv.exe*

files. Instructions for installing both server types are provided in “Installing the Server with the Executable Files” on page 233.

You also need to provide these users with the SentinelSuperPro distributor project file (.DST) containing the protection strategy for the application they will be providing field activation for.

### Opening the License Generator Utility

To open the License Generator utility:

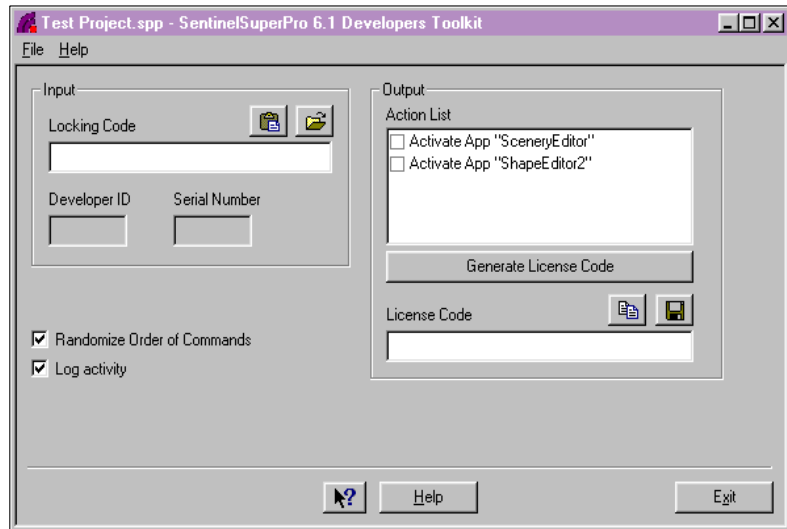
1. Verify the SentinelSuperPro Server is installed and running. See “Verifying the SentinelSuperPro Server Is Running” on page 261 for instructions.
2. Do one of the following:
  - From the **Start** menu, point to **Programs > Rainbow Technologies > SuperPro > 6.1**, then select **License Generator Utility**.
  - Double-click the ***LicenseGenUtil.exe*** file icon.

---

**Note:** *If you did not use the SentinelSuperPro setup program to install the License Generator Utility, it will not be accessible from the Start > Programs menu.*

---





License Generator Utility



## Generating a License Code

Before you can generate a license code, you **must** receive a locking code from your customer for the key protecting the application they want to update. Once you have received the customer's locking code, you are ready to generate a corresponding license code.

To generate a license code:

1. If you are using a distributor key, connect the key to the appropriate port.
2. From the **File** menu, select **Import .DST File** (if you are a distributor) or **Open** (if you are a developer). The Open dialog box appears.
3. Browse to locate the SentinelSuperPro distributor file (.DST) or project file (.SPP) containing the protection strategy for the application you are providing field activation for, then click **Open**.

4. Under **Input**, in the **Locking Code** field, enter the locking code provided by your customer.

- If you copied the locking code to the clipboard, click the paste button  to paste the code in the field.
- If the locking code was saved to a file (.LOC), click the open button  to locate and open the file. The code is entered in the field automatically.

The developer ID and serial number of the key the code was generated from appears in the corresponding fields.

5. Under **Output**, in the **Action** list, select the action(s) you want to perform on the customer's key.

You can select as many actions as necessary. To remove an action, clear the corresponding check box.

---

**Note:** Only selected actions (identified with a check mark) are added to the license code's script. Remember, actions determine what elements of the application will be activated or upgraded. For more information about actions, see Chapter 10, "Defining Field Activation Actions," on page 201.

---

6. If you don't want to randomize the order in which commands are applied to the key in the field, clear the **Randomize Order of Commands** check box.

While randomizing the order commands are applied in can make it harder for hackers to trace what you are doing in code, there are certain situations in which you should *not* randomize commands.

Assume you have two commands that operate on the same cell. One command must be executed before the other because the second command expects the cell to be in a state set by the first command. In this case, you should *not* randomize commands.


Randomizing commands can produce a different license code for a given action set, even if the locking code is the same. Non-randomized commands produce the same license code every time for any given locking code and action set.


7. Select the **Log Activity** check box to create a log file (*LicenseGen.log*) containing all the license codes you generate.

8. Click **Generate License Code**.

The License Generator Utility creates a unique license code based on the locking code and the actions you selected in step 5. The license code appears in the License Code field.

9. If necessary, do one of the following:

- To copy the license code to the clipboard, click the copy button  .

- To save the license code to a file, click the save button  and select a file name and location for the license code file.

10. Refer to “Sending the License Code to Your Customer” on page 255 for instructions on sending the license code to your customer.

## Using the Field Exchange Utility

The Field Exchange Utility is the only stand-alone utility you will send to your customers. An alternative to the Client Activator, it is used to first generate the locking code needed to create a license code, and then enter the license code that performs the field activation commands.

This utility needs to be sent to your customers along with your protected application only if you will be using field activation **and** you are not using the Client Activator. To use the Field Exchange Utility, the SentinelSuperPro Server must also be installed and running on the same workstation.

As a developer, you may want to use this utility as a quick means of testing your keys and the field activation process. After you have completed your protection strategy and programmed some keys, experiment with different field activation commands to make sure the license code applies those commands correctly.

### Installing the Field Exchange Utility – Developers

The Field Exchange utility (*FieldExUtil.exe*) is installed automatically during SentinelSuperPro setup, in the same folder where you installed the SSP Toolkit. No further installation is necessary.

### Installing the Field Exchange Utility – Customers

If you are shipping the Field Exchange utility to your customers for field activation instead of the Client Activator, you should install the Field Exchange executable file (*FieldExUtil.exe*), field exchange .DLL (*usafe32.dll*) and Help file (*fiellexutil.chm*) as part of your application's setup routine.

In order to update keys, the Field Exchange Utility must be installed on the same system where the SentinelSuperPro key has been connected.

- If your application is a stand-alone application, the Field Exchange Utility should be installed on each client system.

- If your application is a network application, the Field Exchange Utility should be installed only on the server where the SentinelSuperPro key is located; it does not need to be installed on the client systems.

Be sure to modify your installation programs appropriately.

---

---

**Warning! DO NOT** send your customers the SentinelSuperPro project (.SPP) containing the protection strategy for the application they purchased.

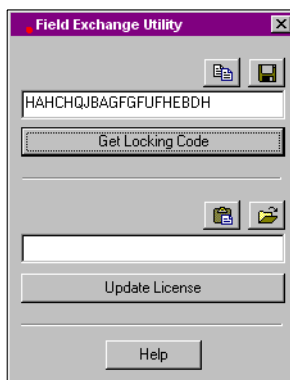
---

---

## Opening the Field Exchange Utility

To open the Field Exchange utility:

1. Verify the SentinelSuperPro Server is installed and running. See “Verifying the SentinelSuperPro Server Is Running” on page 261 for instructions.
2. Do one of the following:
  - If you are a developer, from the **Start** menu, point to **Programs > Rainbow Technologies > SuperPro > 6.1**, then select **Field Exchange Utility**.
  - If you are a user, double-click the **FieldExUtil.exe** file icon.



**Field Exchange Utility Window**

---

**Note:** Because the Field Exchange Utility is designed for use by end users, the instructions in the following two sections are written as if they are being read by an end user. You may want to reproduce these sections and send them to your customers as part of your product’s documentation if you will be providing them with the Field Exchange Utility.

---

## Generating a Locking Code

To update the hardware key used to run your application, you must provide information about the key to your software provider. The Field Exchange Utility displays this information in the form of a locking code, similar to the following:



EHBFGYARDIJABRFLEBDH

You must communicate the locking code to the software provider as they have instructed (for example, via fax or e-mail). The provider will then give you a corresponding license code to enter into the Field Exchange Utility.

To generate a locking code for a key:

1. Verify the correct hardware key is attached to the appropriate port on your computer or server.
2. Click **Get Locking Code**. The locking code appears in the top field.

If the message “Error” appears in the top field, make sure the key is firmly attached to the port and try again.

3. Do one of the following:
  - Click the copy button  to place the locking code on the clipboard.
  - Click the save button  to save the locking code to a file. Define a location and file name for the file, then click **Save**.
  - Write down the code exactly as it appears in the field.
4. Send the locking code to the software provider as directed.
5. Wait to continue the update process until you have received the license code from your software provider.

You can leave the Field Exchange Utility open, or you may close it. Either way, the license code you receive will update your key correctly.

## Entering a License Code

The license code provided to you by the software provider will reprogram the hardware key used to run the application, giving you access to the additional services or features you have purchased. License codes look the same as locking codes.

License codes are unique to the key they were generated for. License codes used with one key cannot be used to activate upgrades for another key, even if the application the key is used with is the same.



To enter a license code:

1. If necessary, open the Field Exchange Utility.
2. Verify the correct key is attached to your computer or server. Only one hardware key should be attached to your computer during the update process.

---

**Note:** *If you are updating a key on a server, you **must** remove all other keys, leaving only the one you are updating. Only one key should be attached to the server during the update process. Because removing keys may terminate the application on some client systems, you may want to perform key updates only during non-peak hours.*

---

3. In the field above the Update License button, enter the license code given to you by your software provider. Do one of the following:
  - Click the paste button  to paste the code from the clipboard, if it was placed there.
  - Click the open button  to load the code from a file. Browse to locate the file (.LIC), then click Open.
  - Type the code in the field. Be sure to enter it exactly as it was provided.
4. Click **Update License**. The key update process begins; it may take up to two or three minutes to complete the process.



5. When “Update Successful” appears, close the Field Exchange Utility.

The application is now ready to use, and you should have access to the additional licenses or new features you purchased.

---

**Note:** *If the update process is not successful, verify the key is securely attached to your computer and try again. If the process is still unsuccessful, or you do not have access to the upgrades you purchased when the process is complete, contact your software provider for assistance.*

---



# Chapter 15

---

## API Function Reference

The SentinelSuperPro API is a set of functions (calls) used to communicate between your application, the network server (when necessary), the Sentinel driver and the hardware key. When you use integrated protection, API functions embedded in your source code call the hardware key to verify its presence, obtain a license and/or perform some other predefined action.

This chapter provides an overview of the SentinelSuperPro API functions, including a list of API status codes. The functions in this chapter are based on the C programming language, and can be used for both stand-alone and network applications.

---

## Using the SentinelSuperPro API

All functions require a pointer to a packet record (RB\_SPRO\_APIPACKET) as a parameter. The Sentinel driver uses the data in the packet record to communicate with the hardware key. The packet record is initialized by the RNBOsproFormatPacket function, and must reside on a DWORD boundary. You must allocate memory for the record and pass an RB\_SPRO\_APIPACKet pointer to all API functions. An application should never modify the data in the packet.

---

**Note:** *Each APIPACKET record is a license request, so there should be a separate APIPACKET record created for each RNBOsproFindFirstUnit call.*

---

The following is an example of a typical API calling sequence:

1. Include the appropriate SuperPro API definition file in your source code.
2. Link your application with the appropriate SuperPro library file.
3. Declare a variable of type RB\_SPRO\_APIPACKET.
4. Format the API packet that you previously declared using the RNBOsproFormatPacket function call.
5. Initialize the packet you declared using the RNBOsproInitialize function call.
6. Add code to set the contact server using the RNBOsproSetContactServer function call.
7. Add code to check for the presence of the key using the RNBOsproFindFirstUnit function call.
8. If you are using sublicensing, make a call to obtain the sublicense using the RNBOsproGetSubLicense function call.

9. Now you can use other SuperPro API functions, such as RNBOsproQuery, RNBOsproRead, etc. as appropriate for the protection scheme you have designed.
10. Before your application terminates, call the RNBOsproReleaseLicense function to release all licenses and sublicenses held by your application.

---

**Note:** *The exact name of the API functions may be different than those listed above, depending on the interface you are using.*

---

Also, make sure you call a SuperPro API function, such as RNBOsproQuery or RNBOsproRead, at least once every 90 seconds after obtaining a license. This will provide the application heartbeat so that the server keeps the license allocated to the application. See “Maintaining the License” on page 106 for more information.

For additional information about using API functions in your protection strategy, refer to Chapter 4, “Designing Your Protection Strategy,” on page 55. Also, for information about setting the application’s access mode through code, see “Setting the Access Mode” on page 100.

---

**Tip:** *Sample code can be found in the pseudocode generated during the Prototype stage. See “Viewing the Pseudocode” on page 192 for more information.*

---

## API Functions Summary

The following table summarizes the SentinelSuperPro API functions. Each function is explained in further detail later in this chapter.

**Summary of API Functions**

Function	See...	Description
RNBOsproActivate()	Page 286	Activates an inactive algorithm so it can be used by the RNBOsproQuery() function.
RNBOsproDecrement()	Page 288	Decrements a counter word or read/write data word by one. If the counter is associated with an active algorithm, decrementing to zero deactivates the algorithm.
RNBOsproEnumServer()	Page 290	Enumerates the number of servers running on the network, according to the specified developer ID.
RNBOsproExtendedRead()	Page 292	Reads the value and access code of any unhidden memory cell in the key.
RNBOsproFindFirstUnit()	Page 293	Searches all attached keys for a specified developer ID.
RNBOsproFindNextUnit()	Page 294	Searches for the next key with the same developer ID.
RNBOsproFormatPacket()	Page 295	Validates the size of the packet (RNBO_SPRO_APIPACKET) and initializes field defaults. <b><i>This function must be called once before any other API function is called.</i></b>
RNBOsproGetContactServer()	Page 296	Returns the contact server set for a particular API packet.
RNBOsproGetFullStatus()	Page 297	Returns extended status information. It is provided for support purposes only.
RNBOsproGetHardLimit()	Page 298	Retrieves the maximum number of licenses supported by the hardware key (the <i>hard limit</i> ).
RNBOsproGetKeyInfo()	Page 299	Gets information about the key from a particular server.
RNBOsproGetSubLicense()	Page 301	Finds a sublicense in a particular cell.

**Summary of API Functions (Continued)**

Function	See...	Description
RNBOsproGetVersion()	Page 302	Returns the SentinelSuperPro driver's version number.
RNBOsproInitialize()	Page 304	Performs any required initialization of the driver.
RNBOsproOverwrite()	Page 305	Changes the value and/or access code of any cell except the reserved cells 00–07.
RNBOsproQuery()	Page 307	Sends a data string to the key, encrypts it using a specified algorithm, and returns the encrypted string to the application.
RNBOsproRead()	Page 310	Reads the value of any unhidden cell in the key.
RNBOsproReleaseLicense()	Page 311	Releases a license by specifying the cell address as zero, or releases a sublicense from a particular cell by specifying the cell address of the sublicensing cell as well as the number of sublicenses to be released.
RNBOsproSetContactServer	Page 312	Sets the contact server for a particular API packet.
RNBOsproWrite()	Page 313	Changes the value and/or access code of any cell with an access code of 0 (read/write data)

## RNBOsproActivate

This function activates an inactive algorithm at the specified address.

If the return value is successful, the algorithm was made active. Error return values will occur if:

- the write password is invalid.
- the activation password is invalid.
- the address is not word 1 of an algorithm having an activation password.

An algorithm can have both a password and counters associated with it. The counters can be used to make the algorithm inactive and the password can be used to make the algorithm active. See **RNBOsproDecrement**.

### Format

```
unsigned short int RNBOsproActivate(
    RB_SPRO_APIPACKET    packet,
    unsigned short int    writePassword,
    unsigned short int    activatePassword1,
    unsigned short int    activatePassword2,
    unsigned short int    address
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>writePassword</i>	The password for the SentinelSuperPro key.
<i>activatePassword1</i>	The first word of the activation password.
<i>activatePassword2</i>	The second word of the activation password.
<i>address</i>	The address of the first word of the inactive algorithm to be activated.



## Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## Additional Information

For more information about using RNBOsproActivate, see “Using Activation Passwords” on page 76.

## RNBOsproDecrement

This function is used to decrement the counter at the specified address.

If the return value is successful, the counter was decremented by one. Error return values will occur if:

- you try to decrement a locked or hidden word.
- the counter is already 0.
- the write password is incorrect.
- the word at the address is not a counter.

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

An algorithm can have both a password and counters associated with it.

The counters can be used to make the algorithm inactive and the password can be used to make the algorithm active. See **RNBOsproActivate**.

### Format

```
unsigned short int RNBOsproDecrement(
    RB_SPRO_APIPACKET      packet,
    unsigned short int      writePassword,
    unsigned short int      address
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>writePassword</i>	The password for the SentinelSuperPro key or the password you define.
<i>address</i>	The address of the counter to decrement.

## Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## Additional Information

For more information about using RNBOsproDecrement, see “Controlling Demo Applications” on page 90.

## RNBOSproEnumServer

This function is used to enumerate the number of servers running on the network, according to the specified developer ID.

### Format

```
unsigned short int SP_API RNBOSproEnumServer(
    ENUM_SERVER_FLAG      enumFlag,
    unsigned short int     developerID,
    NSPRO_SERVER_INFO      *serverInfo,
    unsigned short int     *numServerInfo
);
```

### Parameters

<i>enumFlag</i>	The flag that tells whether to contact one server (NSPRO_RET_ON_FIRST) or all servers (NSPRO_GET_ALL_SERVERS).
<i>developerID</i>	The developer ID. Only servers that have a key with this developer ID attached will respond; the server response will include the total number of licenses available for the key ( <i>numLicAvail</i> ). If the developer ID is specified as 0xFFFF, all the servers in the subnet will respond.
<i>serverInfo</i>	A pointer to a buffer large enough to hold an NSPRO_SERVER_INFO structure for each server enumerated.

```
#define    MAX_ADDR_LEN 32
typedef struct {
    char                serverAddress[MAX_ADDR_LEN];
    unsigned short int  numLicAvail;
} NSPRO_SERVER_INFO;
```

Be sure to allocate enough room in your buffer to hold the server information you are getting; the more servers you are requesting information for, the larger your buffer should be.

*numServerInfo* A pointer to the variable that will contain the number of servers accessed for information. Before calling this function, set the variable to the number of servers you want information for. When the function returns, the variable will contain the number of servers that information is available for in the *serverInfo* buffer.

## Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## Additional Information

For more information about using RNBOsproEnumServer, see “Finding Additional Servers” on page 104.

## RNBOSproExtendedRead

This function reads the word and access code at the specified address.

On success, the data variable contains the information from the SentinelSuperPro key and the access code variable contains the access code. If the error code is SP\_ACCESS\_DENIED, an attempt was made to read a non-readable word or an algorithm/hidden word. For security reasons, algorithm words cannot be read.

### Format

```
unsigned short int RNBOSproExtendedRead(
    RB_SPRO_APIPACKET    packet,
    unsigned short int    address,
    unsigned short int    *data,
    unsigned char         *accessCode
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>address</i>	The address to be read.
<i>data</i>	A pointer to the variable that will contain the data read from the SentinelSuperPro key.
<i>accessCode</i>	A pointer to the variable that will contain the access code associated with the word that was read.

### Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## RNBOsproFindFirstUnit

This function finds the first SentinelSuperPro key with the specified developer ID. After calling **RNBOsproFormatPacket** and **RNBOsproInitialize()**, you must call this function before using any other calls.

This function essentially acts as a “get license” call. If the SentinelSuperPro key is found, the RB\_SPRO\_APIPACKET record will contain valid license data, otherwise, the packet will be marked invalid. If you try to call this function on an APIPACKET that already has a license, the SP\_INVALID\_OPERATION error is returned.

### Format

```
unsigned short int RNBOsproFindFirstUnit (
    RB_SPRO_APIPACKET    packet,
    unsigned short int    developerID
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>developerID</i>	This is assigned to you by Rainbow Technologies or your distributor. It identifies the SentinelSuperPro device to search for.

### Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

### Additional Information

For more information about using RNBOsproFindFirstUnit, see “Dealing With Missing Hardware Keys” on page 78.

## RNBOsproFindNextUnit

This function finds the next SentinelSuperPro key based on the developer ID maintained in the RB\_SPRO\_APIPACKET record.

This function should not be called unless RNBOsproFindFirstUnit has returned a successful value. If this function returns success, the RB\_SPRO\_APIPACKET record will contain the data for the next SentinelSuperPro key.

RNBOsproFindNextUnit looks for the next key only on the server that was originally contacted by RNBOsproFindFirstUnit; it does not look for keys on other servers.

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 RNBOsproFindFirstUnit and optionally, RNBOsproFindNextUnit, depending upon the number of SentinelSuperPro keys found and which one your application wants to access.

### Format

```
unsigned short int RNBOsproFindNextUnit(
    RB_SPRO_APIPACKET packet
);
```

### Parameters

*packet*                      A pointer to the RB\_SPRO\_APIPACKET record.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.



## RNBOsproFormatPacket

This function initializes and validates the size of the RB\_SPRO\_APIPACKET record that is passed. ***This function must be called before any other API function.***

### Format

```
unsigned short int RNBOsproFormatPacket (
    RB_SPRO_APIPACKET    packet,
    unsigned short int    packetLen
);
```

### Parameters

<i>packet</i>	A pointer to a DWORD-aligned RB_SPRO_APIPACKET record.
<i>packetLen</i>	An integer containing the length of the RB_SPRO_APIPACKET record in bytes.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

---

## RNBOSproGetContactServer

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

### Format

```
unsigned short int SP_API RNBOSproGetContactServer(  
    RBP_SPRO_APIPACKET    packet,  
    char                   *serverNameBuf,  
    unsigned short int     serverNameBufSz  
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>serverNameBuf</i>	A pointer to the buffer where the server name is copied.
<i>serverNameBufSz</i>	The length of the buffer in bytes.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

---

## RNBOsproGetFullStatus

This function is used to return extended status information. It is provided for support purposes only.

### Format

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

### Parameters

*packet*                      A pointer to the RB\_SPRO\_APIPACKET record.

### Return Values

Returns an RB\_WORD value that can be interpreted by Rainbow's technical support department.

---

## RNBOSproGetHardLimit

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

### Format

```
unsigned short int RNBOSproGetHardLimit(  
    RBP_SPRO_APIPACKET    packet  
    unsigned short int     *HardLimit  
);
```

### Parameters

*packet*                      A pointer to the RB\_SPRO\_APIPACKET record.

*HardLimit*                 A pointer to the buffer that will hold the hard limit.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## RNBOsproGetKeyInfo

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

### Format

```
unsigned short int  RNBOsproGetKeyInfo (
                    RBP_SPRO_APIPACKET    packet,
                    unsigned short int     devId,
                    unsigned short int     keyIndex,
                    NSPRO_MONITOR_INFO     *nsproMonitorInfo
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>devId</i>	The developer ID of the key (at the keyIndexth position) being accessed.
<i>keyIndex</i>	The keyIndex of the key whose information is being sought.
<i>numMonitorInfo</i>	A pointer to the MonInfo structure. This structure has various fields that contain information about the key.

```
#define    MAX_ADDR_LEN    32
#define    MAX_NAME_LEN    64
typedef struct tag_nsproKeyMonitorInfo {
    unsigned short int     devId;
    unsigned short int     hardLimit;
    unsigned short int     inUse;
    unsigned short int     numTimeOut;
    unsigned short int     highestUse;
} NSPRO_KEY_MONITOR_INFO;
```

```
typedef struct tag_nsproMonitorInfo {
    char                serverName[MAX_NAME_LEN];
    char                serverIPAddress[MAX_ADDR_LEN];
    char                serverIPXAddress[MAX_ADDR_LEN];
    char                version[MAX_NAME_LEN];
    unsigned short int  protocol;
    NSPRO_KEY_MONITOR_INFO sproKeyMonitorInfo;
} NSPRO_MONITOR_INFO;
```

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

---

## RNBOsproGetSubLicense

This function obtains a sublicense in a particular cell.

### Format

```
unsigned short int  RNBOsproGetSubLicense (
                    RBP_SPRO_APIPACKET    packet,
                    unsigned short int     address
                ) ;
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>address</i>	The cell address where the sublicense will be obtained.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

### Additional Information

For more information about using RNBOsproGetSubLicense, see “Getting a Sublicense” on page 108.

---

## RNBOSproGetVersion

This function returns the driver's version and type.

### Format

```
unsigned short int RNBOSproGetVersion(  
    RB_SPRO_APIPACKET    packet,  
    unsigned char         *majVer,  
    unsigned char         *minVer,  
    unsigned char         *rev,  
    unsigned char         *drvType  
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>majVer</i>	A pointer to the location for the returned major version number.
<i>minVer</i>	A pointer to the location for the returned minor version number.
<i>rev</i>	A pointer to the location for the returned revision number.
<i>drvType</i>	<div>A pointer to the location for the returned driver type identifier. Current driver types:<ul style="list-style-type: none"><li>• RB_DOSRM_LOCAL_DRV (1): DOS local driver</li><li>• RB_WIN3x_LOCAL_DRV (2): Windows 3.x local driver</li><li>• RB_WIN32s_LOCAL_DRV (3): Windows Win32s local driver</li><li>• RB_WIN3x_SYS_DRV (4): Windows 3.x system driver</li><li>• RB_WINNT_SYS_DRV (5): Windows NT system driver</li><li>• RB_OS2_SYS_DRV (6): OS/2 system driver</li></ul></div>



- RB\_WIN95\_SYS\_DRV (7): Windows 95 system driver
- RB\_NW\_LOCAL\_DRV (8): NetWare local driver
- RB\_QNX\_LOCAL\_DRV (9): QNX local driver

## Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

---

## RNBosproInitialize

This function initializes the SentinelSuperPro device and the RB\_SPRO\_APIPACKET that is passed. It allows the driver to perform any needed initialization and sets the contact server if anything is set in the NSP\_HOST environment variable. Your application must call this function one time before calling any other API functions.

### Format

```
unsigned short int RNBosproInitialize(  
    RB_SPRO_APIPACKET    packet  
) ;
```

### Parameters

*packet*                      A pointer to the RB\_SPRO\_APIPACKET record.

### Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## RNBOsproOverwrite

This function allows the application to change the value and access code of any word of in the SentinelSuperPro key.

Use of this function requires knowledge of the write password and the overwrite passwords. The word data is placed in the data variable and its associated access code in the access code variable.

On success, the data and its associated access code were written to the specified word on the SentinelSuperPro key. If the error code is SP\_ACCESS\_DENIED, either the write password or the overwrite passwords were incorrect.

This function can be used to overwrite any word on the SentinelSuperPro key with the exception of the words at addresses 00 through 07.

### Format

```
unsigned short int RNBOsproOverwrite (
    RB_SPRO_APIPACKET      packet,
    unsigned short int      writePassword,
    unsigned short int      overwritePassword1,
    unsigned short int      overwritePassword2,
    unsigned short int      address,
    unsigned short int      data,
    unsigned      char      accessCode
);
```

### Parameters

*packet*                      A pointer to the RB\_SPRO\_APIPACKET record.

*writePassword*            The write password for the SentinelSuperPro key.

*overwritePassword1*      The overwrite password word 1.

<i>overwritePassword2</i>	The overwrite password word 2.
<i>address</i>	The address of the word to write. It is valid for any cell except for cell addresses 00 through 07.
<i>data</i>	Contains the SentinelSuperPro word to write.
<i>accessCode</i>	Contains the access code associated with the word to write.

**Return Values**

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## RNBOsproQuery

This function is used to query an active algorithm at the specified address. The address should be the first word of the active algorithm.

The query data pointer will point to the first byte of the data to be passed to the active algorithm. The length of the query data is specified in the length variable.

On success, the query response of the same length will be placed in the buffer pointed to by the response pointer. The last four bytes of the response will also be placed in the Response32 variable.

Each query byte may contain any value from 0 to 255. Each response byte may also contain any value from 0 to 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 the memory for the buffers.

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

### Format

```
unsigned short int RNBOsproQuery(
    RB_SPRO_APIPACKET    packet,
    unsigned short int    address,
    VOID                  *queryData,
    VOID                  *response,
    unsigned long          *response32,
    unsigned short int     length
);
```

## Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>address</i>	The address of the word to query.
<i>queryData</i>	The pointer to the first byte of the query bytes.
<i>response</i>	The pointer to the first byte of the response bytes.
<i>response32</i>	The pointer to the location that will contain a copy of the last four bytes of the query response.
<i>length</i>	This is the number of query bytes to send to the active algorithm and also the length of the response buffer.

## Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## Additional Information

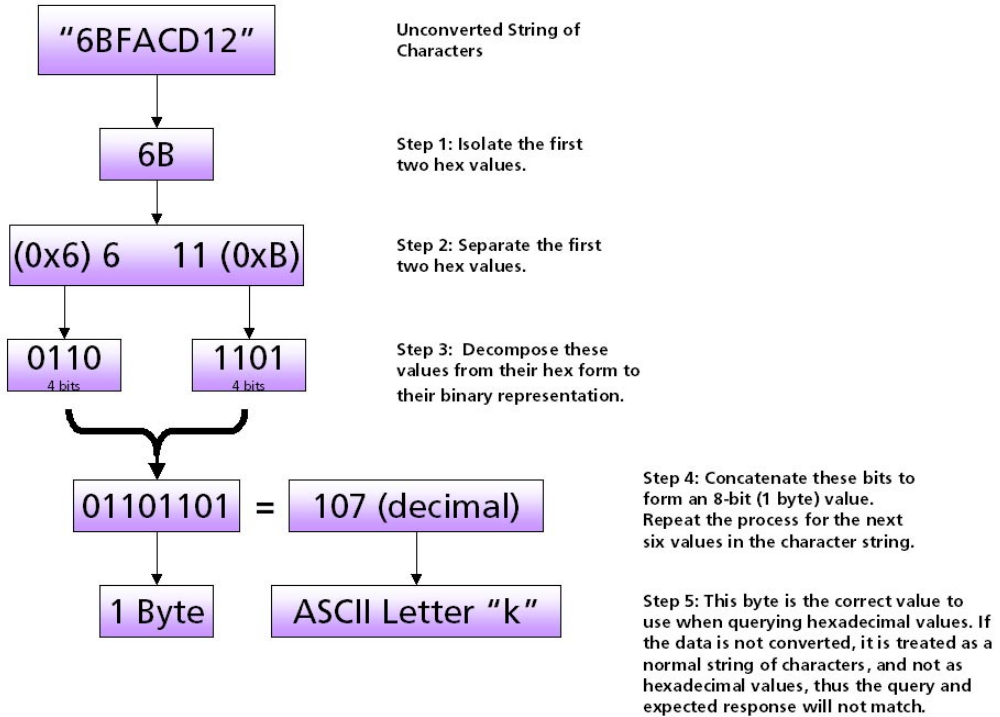
The diagram on the following page is designed to help you understand how the RNBOsproQuery function accepts and returns values, and how you should convert these values.

## Additional Information

For more information about using RNBOsproQuery, see the following:

- “Using Activation Passwords” on page 76
- “Advanced Protection Techniques” on page 81
- “Querying Counters” on page 95

## Sample Conversion from Hexadecimal to Decimal



## RNBOSproRead

This function reads a word at the specified address.

If successful, the data variable will contain the word's value. If the error code is `SP_ACCESS_DENIED`, an attempt was made to read a non-readable word or algorithm/hidden word. For security reasons, algorithm words cannot be read.

### Format

```
unsigned short int RNBOSproRead(
    RB_SPRO_APIPACKET    packet,
    unsigned short int    address,
    unsigned short int    *data
);
```

### Parameters

<i>packet</i>	A pointer to the <code>RB_SPRO_APIPACKET</code> record.
<i>address</i>	The SentinelSuperPro key memory cell address of the word to read.
<i>data</i>	A pointer to the location that will contain the data read from the SentinelSuperPro key.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.



## RNBOsproReleaseLicense

This function can either release a license by specifying the cell address as zero, or it can release a sublicense from a particular cell by specifying the cell address of the sublicensing cell as well as the number of sublicenses to be released. To free up memory, use this function to release all licenses and sublicenses before your application terminates.

### Format

```
unsigned short int  RNBOsproReleaseLicense(
    RBSPRO_APIPACKET    packet,
    unsigned short int    address,
    unsigned short int    *numSubLic
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>address</i>	The cell address (if a sublicense is to be released, specify the sublicense cell number, else send 0).
<i>numSubLic</i>	The pointer to the variable containing the number of sublicenses to be released. If a license is to be released, this can be specified as null.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

---

## RNBOsproSetContactServer

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

This function will not work if the API packet already has a license. Instead, it will return `SP_INVALID_OPERATION`.

### Format

```
unsigned short int RNBOsproSetContactServer(  
    RBP_SPRO_APIPACKET    packet,  
    char                   *serverName  
);
```

### Parameters

<i>packet</i>	A pointer to the <code>RB_SPRO_APIPACKET</code> record.
<i>serverName</i>	The name to which the contact server is set.

### Return Values

If successful, the function returns **SP\_SUCCESS(0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

### Additional Information

For more information about using `RNBOsproSetContactServer`, see “Setting Stand-alone or Network Mode” on page 100.

## RNBOsproWrite

This function is used to write a word and its associated access code to a specified address.

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

If successful, the data and its associated access code were written to the specified word on the SentinelSuperPro key. If the error code is SP\_ACCESS\_DENIED, either the write password was incorrect, or an attempt was made to overwrite a locked word on the SentinelSuperPro key.

The write function can only be used to overwrite words with an access code of 0 in the SentinelSuperPro key. To overwrite words with other access codes, use the **RNBOsproOverwrite** function.

### Format

```
unsigned short int RNBOsproWrite(
    RB_SPRO_APIPACKET      packet,
    unsigned short int      writePassword,
    unsigned short int      address,
    unsigned short int      data,
    unsigned char           accessCode
);
```

### Parameters

<i>packet</i>	A pointer to the RB_SPRO_APIPACKET record.
<i>writePassword</i>	The write password for the SentinelSuperPro key.
<i>address</i>	The address of the word to write.
<i>data</i>	Contains the SentinelSuperPro word to write.

<i>accessCode</i>	Contains the access code associated with the word to write.
-------------------	---

### Return Values

If successful, the function returns **SP\_SUCCESS (0)**. If an error occurs, the function returns one of the status codes listed in “API Status Codes” on page 315.

## API Status Codes

The following table describes the status codes an API function can return to your application.

**API Function Status Codes**

Status Code (Decimal)	Description
0	<b>SP_SUCCESS</b> The function completed successfully.
1	<b>SP_INVALID_FUNCTION_CODE</b> You specified an invalid function code. See your language's include file (for example, SUPERPRO.H) for valid API function codes. Generally, this error should not occur if you are using a Rainbow-provided interface to communicate with the driver. However, may occur when a stand-alone-only function is used in a network situation.
2	<b>SP_INVALID_PACKET</b> A checksum error was detected in the command packet, indicating an internal inconsistency. The packet record has not been initialized, or may have been tampered with. Generally, this error should not occur if you are using a Rainbow-provided interface to communicate with the driver.
3	<b>SP_UNIT_NOT_FOUND</b> Either RNBOsproFindFirstUnit or RNBOsproFindNextUnit could not find the specified SentinelSuperPro key. Make sure you are sending the correct developer ID. This error is returned by other functions if the key has been removed.
4	<b>SP_ACCESS_DENIED</b> You attempted to perform an illegal action on a cell. For example, you may have tried to read an algorithm/hidden cell, write to a locked cell, or decrement a cell that is not a data or counter word.

## API Function Status Codes (Continued)

Status Code (Decimal)	Description
5	<b>SP_INVALID_MEMORY_ADDRESS</b> You specified an invalid SentinelSuperPro memory address. Valid addresses are 0–63 decimal (0–3F hex). Cells 00–07 are invalid for many operations. Algorithms must be referenced by the first (even) address.
6	<b>SP_INVALID_ACCESS_CODE</b> 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	<b>SP_PORT_IS_BUSY</b> The requested operation could not be completed because the port is busy. This can occur if there is considerable printer activity, or if a unit on the port is performing a write operation and is blocking the port. Try the function again.
8	<b>SP_WRITE_NOT_READY</b> The write or decrement could not be performed due to a momentary lack of sufficient power. Try the function again.
9	<b>SP_NO_PORT_FOUND</b> No parallel ports could be found on the local workstation or network server, or there was a problem with the protocol being used on the network.
10	<b>SP_ALREADY_ZERO</b> You tried to decrement a counter or data word that already contains the value zero. If you are using the counter to control demo application executions, this condition may occur after the corresponding algorithm has been reactivated with its activation password.
12	<b>SP_DRIVER_NOT_INSTALLED</b> The system device driver was not installed or detected. Communication to the unit was not possible. Verify the device driver is correctly installed.

## API Function Status Codes (Continued)

Status Code (Decimal)	Description
13	<b>SP_COMMUNICATIONS_ERROR</b> The system device driver is having problems communicating with the unit or the network server. Verify the device driver is correctly installed.
18	<b>SP_VERSION_NOT_SUPPORTED</b> The current system device driver is outdated. Update the driver.
19	<b>SP_OS_ENVIRONMENT_NOT_SUPPORTED</b> The operating system or environment is not supported by the client library. Contact Technical Support for assistance.
20	<b>SP_QUERY_TOO_LONG</b> You sent a query string longer than 56 characters. Send a shorter string.
30	<b>SP_DRIVER_IS_BUSY</b> The system driver is busy. Try the function again.
31	<b>SP_PORT_ALLOCATION_FAILURE</b> Failure to allocate a parallel port through the operating system's parallel port contention handler.
32	<b>SP_PORT_RELEASE_FAILURE</b> Failure to release a previously allocated parallel port through the operating system's parallel port contention handler.
39	<b>SP_ACQUIRE_PORT_TIMEOUT</b> Failure to acquire access to a parallel port within the defined time-out.
42	<b>SP_SIGNAL_NOT_SUPPORTED</b> The particular machine does not support a signal line. For example, an attempt may have been made to use the ACK line on an NEC 9800 computer.
57	<b>SP_INIT_NOT_CALLED</b> The key is not initialized. Call the RNBOsproInitialize function before calling the function that generated this error.

## API Function Status Codes (Continued)

Status Code (Decimal)	Description
58	<b>SP_DRIVER_TYPE_NOT_SUPPORTED</b> The type of driver access, either direct I/O or system driver, is not supported for the defined operating system and client library.
59	<b>SP_FAIL_ON_DRIVER_COMM</b> The client library failed on communicating with the Rainbow system driver.
60	<b>SP_SERVER_PROBABLY_NOT_UP</b> The server is not responding; the client has timed out.
61	<b>SP_UNKNOWN_HOST</b> The server host is unknown. The server is not on the network, or an invalid host name was specified.
62	<b>SP_SENDTO_FAILED</b> The client was unable to send a message to the server.
63	<b>SP_SOCKET_CREATION_FAILED</b> Client was unable to open a network socket. Make sure the TCP/IP or IPX protocol stack is properly installed on the system.
64	<b>SP_NORESOURCES</b> Could not locate enough licensing requirements. Insufficient resources (such as memory) are available to complete the request, or an error occurred in attempting to allocate needed memory.
65	<b>SP_BROADCAST_NOT_SUPPORTED</b> Broadcast is not supported by the network interface on the system.
66	<b>SP_BAD_SERVER_MESSAGE</b> Could not understand a message received from the server. An error occurred in decrypting (or decoding) a server message at the client end.
67	<b>SP_NO_SERVER_RUNNING</b> Cannot talk to server. Verify the server is running. Server on specified host may not be available for processing the request.



## API Function Status Codes (Continued)

Status Code (Decimal)	Description
68	<b>SP_NO_NETWORK</b> Unable to talk to the specified host. Network communication problems encountered.
69	<b>SP_NO_SERVER_RESPONSE</b> No server responded to the client broadcast. There is no server running in the subnet, or no server in the subnet has the desired key connected.
70	<b>SP_NO_LICENSE_AVAILABLE</b> All licenses are currently in use. Server has no more licenses available for this request.
71	<b>SP_INVALID_LICENSE</b> License is no longer valid. License expired due to time-out.
72	<b>SP_INVALID_OPERATION</b> Specified operation cannot be performed. Tried to set the contact server after obtaining a license, or tried to call FindFirstUnit function on a packet that already has a license.
73	<b>SP_BUFFER_TOO_SMALL</b> The size of the buffer is not sufficient to hold the expected data.
74	<b>SP_INTERNAL_ERROR</b> An internal error, such as failure to encrypt or decrypt a message being sent or received, has occurred.
75	<b>SP_PACKET_ALREADY_INITIALIZED</b> The packet being initialized was already initialized.
255	<b>SP_INVALID_STATUS</b> An invalid status code was returned.



# Chapter 16

---

## Migrating From SentinelSuperPro 5.1 or NetSentinel

If you have used a previous version of SentinelSuperPro—such as SentinelSuperPro 5.1 or SentinelSuperPro Advantage—or NetSentinel, read this chapter to find out how features and functions in the older versions have been combined in SentinelSuperPro 6.1.

Applications you protected using SentinelSuperPro 5.1 or earlier will continue to run without problems. If the platform is supported by SentinelSuperPro 6.1, you can edit existing protection strategies by importing your .DAT file. See “Importing a .DAT File” on page 132 for more information.

---

**Note:** *The SentinelSuperPro 6.1.1 Toolkit can reside concurrently with older versions of SentinelSuperPro (5.1 or 6.0) on the same system. Installing the SSP 6.1.1 Toolkit does not overwrite existing SentinelSuperPro applications, such as SAFE or the SMU. However, you **must** uninstall SentinelSuperPro 6.1.0 before you can install SentinelSuperPro 6.1.1.*

---

## **Finding 5.1 Features in SentinelSuperPro 6.1**

Previously, the following applications made up the SentinelSuperPro package:

- SentinelWizard
- SentinelShell
- SentinelSuperPro Advanced Editor
- SentinelSuperPro SAFE
- SentinelSuperPro Manufacturing Utility (SMU)
- Sentinel Evaluation Program

Now, these six applications are consolidated into one easy-to-use application: the SentinelSuperPro 6.1 Toolkit. You no longer have to switch applications, or import and export data, to design and implement your protection strategy.

The table on the following page describes where you can find each of these applications' functions and features in the new version of SentinelSuperPro.

**Where SentinelSuperPro 5.1 Features Are Located in 6.1**

<b>5.1 Feature / Terminology</b>	<b>6.1 Feature / Terminology</b>	<b>For More Information, See...</b>
Windows 3.x / DOS / OS/2 support	No longer available. SentinelSuperPro 6.1 supports Win32 only. If you need 16-bit support, continue using SentinelSuperPro 5.1.	Rainbow Technologies Technical Support. See page xxiii for contact information.
Drop-In Activation Type <i>and</i> Drop-In Keys	Not currently available. If you need to use this activation type, continue using SentinelSuperPro 5.1.	"Activation Types" on page 60
Distributor keys	The Distributor Keys tab in the Make Keys stage.	"Programming a Distributor Key" on page 223
SAFE commands: <ul style="list-style-type: none"> <li>• Read Cell</li> <li>• Check Algo</li> <li>• Check Counter</li> </ul>	Not available in this version.	Chapter 10, "Defining Field Activation Actions," on page 201
SAFE_CFG application	The DLLs used for field activation and exchange are now created when you open the SSP Toolkit and enter your developer ID and passwords for the first time.	Chapter 6, "Starting the SentinelSuperPro Toolkit," on page 111
SentinelShell	"Shelled" application protection is available from the Design stage.	"Using Automatic Protection" on page 152
Sentinel Manufacturing Utility (SMU)	Make Keys stage or Make Keys Utility	Chapter 11, "Programming Keys," on page 213
USAFE	Field Exchange Utility	Chapter 14, "Using the Stand-alone Utilities," on page 259
DSAFE	License Code Generator in the Implementation stage, or the stand-alone License Generator Utility	Chapter 13, "Activating and Updating Keys," on page 243

**Where SentinelSuperPro 5.1 Features Are Located in 6.1 (Continued)**

5.1 Feature / Terminology	6.1 Feature / Terminology	For More Information, See...
Key ID String	Locking Code	"What Is a Locking Code?" on page 245
Update Key String	License Code	"What Is a License Code?" on page 246
Defining update actions	Field Activation tab of the Implementation stage	Chapter 10, "Defining Field Activation Actions," on page 201
DAT files	No longer created by SentinelSuperPro. Protection strategies are now saved in a SentinelSuperPro Project file (.SPP). However, .DAT files created by an older version may be imported into a SentinelSuperPro 6.1 project file.	"What Is a Project?" on page 131 and "Importing a .DAT File" on page 132.
Advanced Editor	Creating custom elements in the Design stage.	Chapter 8, "Working With Design Elements," on page 169.
Driver encryption / SPROX	Because Win 3.x and DOS are no longer supported, driver encryption is no longer necessary.	
SentinelWizard	Has been folded into the SentinelSuperPro interface. The Element Definition Wizard, accessible via the Design stage, performs a similar function.	Chapter 7, "Protecting Your Application," on page 141.

---

**Tip:** For a list of the new features in SentinelSuperPro 6.1, see "What's New in SentinelSuperPro 6.1?" on page 14.

---

## Finding NetSentinel Features in SentinelSuperPro 6.1

Previously, the following applications made up the NetSentinel package:

- NetSentinel Server
- NetSentinel Monitor
- NetSentinel Editor

Now, these applications are consolidated into the SentinelSuperPro 6.1 Toolkit. The following table describes where you can find each of these applications' functions and features in the new version of SentinelSuperPro.

**Where NetSentinel Features Are Located in SentinelSuperPro 6.1**

NetSentinel Feature / Terminology	6.1 Feature / Terminology	For More Information, See...
Changeable license limits	Provided through the use of sublicensing.	"Using Sublicenses" on page 107
Soft license limits	License control is provided to developers only through the use of sublicensing.	"Using Sublicenses" on page 107
Local and network access	Access modes: stand-alone, network or dual.	"Setting the Access Mode" on page 100
Department names	No longer required or supported.	
Multiple servers	More than one key can be connected per server, and more than one server can be on the network.	"Installing the SentinelSuperPro Hardware Key" on page 27
Shared licensing	No longer supported.	
DOS–Windows–OS/2 connectivity	No longer supported. Win32 only is supported.	
Reporting and monitoring	The SentinelSuperPro Server log file and the SentinelSuperPro Monitoring Tool.	<i>The SentinelSuperPro System Administrator's Guide</i>





# Appendix A

---

## Using the Command Line Shell Utility

If you want to protect your application while you are still developing it, you can use the Shell utility to easily “shell” your application after each build. This utility is command line-based.

To use the Shell utility to add automatic (“shelled”) protection to your application during the build process, add a batch file that calls an existing SentinelSuperPro project. The automatic protection options you set up and saved in the SSP Toolkit are then used to shell your application during the build process.

---

**Note:** A SentinelSuperPro project file (\*.spp) that defines the automatic (“shelled”) protection options for the application(s) you want to shell **must** exist before you can use the Shell utility. For more information about setting up automatic protection, see Chapter 7, “Protecting Your Application,” on page 141.

---

In previous versions of the SentinelShell command line, you could specify shell protection options, such as adding execution control or identifying files for encryption. In this version, there are no design-related command line options; all design options must be set up in the SSP Toolkit and saved to a project file. See “Using Automatic Protection” on page 152 for instructions.

## Command Line Syntax

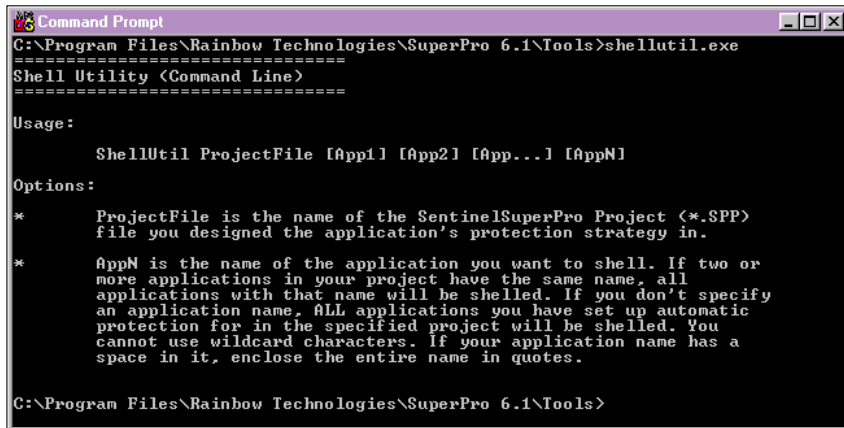
The syntax for shelling an application from the command line is:

```
ShellUtil ProjectFile [App 1] [App 2]...[App N]
```

where:

- **ShellUtil** is the name of the command line shell utility
- **Project File** is the name of the SentinelSuperPro project file (\*.spp) you designed the application's protection strategy in.
- **[App 1]** is the name of the application you want to shell, as it was specified in the SentinelSuperPro project.

More than one application may be shelled with the same command. See "Using the Shell Utility" on page 329 for more information.



```

Command Prompt
C:\Program Files\Rainbow Technologies\SuperPro 6.1\Tools>shellutil.exe
=====
Shell Utility <Command Line>
=====
Usage:
    ShellUtil ProjectFile [App1] [App2] [App...] [AppN]
Options:
*   ProjectFile is the name of the SentinelSuperPro Project (*.SPP)
    file you designed the application's protection strategy in.
*   AppN is the name of the application you want to shell. If two or
    more applications in your project have the same name, all
    applications with that name will be shelled. If you don't specify
    an application name, ALL applications you have set up automatic
    protection for in the specified project will be shelled. You
    cannot use wildcard characters. If your application name has a
    space in it, enclose the entire name in quotes.
C:\Program Files\Rainbow Technologies\SuperPro 6.1\Tools>
  
```

### Command Prompt – Shell Utility

---

## Using the Shell Utility

When using the Shell utility command line, keep in mind the following:

- If you don't specify an application name in the command line, *all* applications you have set up automatic protection for in the specified project will be shelled.
- If two or more applications have the same name in your project, and you specify that name on the command line, *all* applications with that name will be shelled.
- If the name of your application has a space in it, you must enclose the entire name in quotes. For example, if the name of your application is **Project Manager**, you would enter "**Project Manager**" on the command line.
- You cannot use wildcard characters.
- If using the command line in a build process, shelling an application is a post-build step that should occur after the link stage.

### Example

The following example shows how to use the Shell utility to apply automatic protection to one or more applications.

Assume you have a SentinelSuperPro project file named *MyApps.spp*. In this project, you added three applications—**ShapeEditor**, **SceneryEditor** and **TextEditor**.

You designed your protection strategy to use integrated protection for ShapeEditor, and automatic protection for SceneryEditor and TextEditor.

To shell SceneryEditor and/or TextEditor using the command line:

1. Open a command prompt and navigate to the directory where the *ShellUtil.exe* is located.

In our example, we navigated to:

```
C:\SentinelSuperPro6\
```

2. Enter the shell command, specifying the SentinelSuperPro project name (including the file extension) and the name of the application(s) you want to shell.

In our example, we wanted to shell only the SceneryEditor application, so we entered:

```
ShellUtil MyApp.spp SceneryEditor
```

If we had entered:

```
ShellUtil MyApp.spp
```

both SceneryEditor and TextEditor would have been shelled. This is because we did not specify an application name, so **all** applications in the project defined as having automatic protection would have been shelled.

In both cases, ShapeEditor would not be shelled because it was defined as having integrated protection and is thus ignored by the Shell utility.

3. Press **Enter**. The applications are shelled via the command line.

# Appendix B

---

## Troubleshooting

Rainbow Technologies is committed to providing easy-to-use products that increase productivity and offer top-quality performance. However, if you do experience a problem, you may save yourself time by first looking over the information and recommendations in this troubleshooting appendix.

This appendix presents common SentinelSuperPro problems and solutions. If you cannot quickly solve your problem using the information in this appendix, please contact Rainbow Technologies Technical Support for further assistance. Contact information can be found on page xxiii.

This appendix covers the following topics:

- Uninstalling the SentinelSuperPro Toolkit
- Repairing a SentinelSuperPro Installation
- Strategy design issues
- Application protection issues
- SentinelSuperPro key compatibility issues
- Key programming issues

---

## Uninstalling the SentinelSuperPro Toolkit

This section explains how to uninstall the SentinelSuperPro Developer's Toolkit and related SentinelSuperPro components from your system.

1. From the **Start** menu, select **Settings > Control Panel**. The Control Panel window appears.
2. Double-click **Add/Remove Programs**. The Add/Remove Programs Properties dialog box appears.
3. From the list of installed applications, select **SentinelSuperPro 6.1**.
4. Click **Add/Remove**. The SentinelSuperPro Installation Wizard Welcome screen appears.
5. Click **Next**. The Modify, Repair or Remove screen appears.
6. Select **Remove**, then click **Next**.
7. You are asked to confirm the removal of SentinelSuperPro 6.1. Click **Remove**. SentinelSuperPro Toolkit uninstallation begins.
8. If any SentinelSuperPro files are in use, the Files in Use screen appears. Close any listed files, then click **Retry**.
9. When uninstallation is complete, the Finished screen appears. Click **Finish**. You are returned to the Add/Remove Programs Properties dialog box.

---

**Note:** Depending on your operating system, you may need to reboot your system at this point. You will be prompted if a reboot is required; if a message appears, follow the on-screen instructions.

---

To completely remove SentinelSuperPro files from your system, you should also uninstall the following components:

- Client Activator
- Client Activator Wizard
- Sentinel System Driver

Repeat steps 3 – 9 above for each of these components, if necessary. When you have finished uninstalling SentinelSuperPro components, click **OK** to close the Add/Remove Programs Properties dialog box.

---

**Note:** *If you have more than one application that uses the Sentinel driver installed on your system, the SentinelSuperPro uninstall process will not uninstall the driver. The driver will be uninstalled only when the last application using the driver is uninstalled. This holds true for the SentinelSuperPro server also.*

---

Also, the following files do not get automatically removed when you uninstall SentinelSuperPro. You should delete these files manually to complete the uninstallation process.

- Log files generated by the SentinelSuperPro server
- Log files generated during key programming or license code generation
- Intermediate files created by compiling interfaces
- Any SentinelSuperPro project files (.SPP or .DST)
- *usafe32.dll* and *dsafe32.dll*

---

## Repairing a SentinelSuperPro Installation

The SentinelSuperPro Installation Wizard features a repair function, allowing you to fix any errors that may have occurred during installation, or reinstall any critical files you may have accidentally deleted or that have become corrupt.

You may need to use the repair function if any of the following occurs:

- Your SentinelSuperPro shortcuts (on the desktop or in the Start menu) stop working
- When you start the SSP Toolkit, a message appears saying you are missing .DLL files
- The SentinelSuperPro Server won't start
- The Sentinel driver appears to be missing

When you run a repair operation, the Installation Wizard locates all the files installed with SentinelSuperPro, and then updates or replaces them as necessary. It also restores any registry entries and shortcuts that were created during the initial installation process.

While the repair function can be a quick and easy way to fix minor problems, it may not always solve all problems. If, after running the repair function, you are still having problems, you should first remove, and then reinstall SentinelSuperPro.

To run a repair operation:

1. From the **Start** menu, select **Settings > Control Panel**. The Control Panel window appears.
2. Double-click **Add/Remove Programs**. The Add/Remove Programs Properties dialog box appears.
3. From the list of installed applications, select **SentinelSuperPro 6.1**.



4. Click **Add/Remove**. The SentinelSuperPro Installation Wizard Welcome screen appears.
5. Click **Next**. The Modify, Repair or Remove screen appears.
6. Select **Repair**, then click **Next**.  
The Ready to Repair screen appears.
7. Click **Install** to start the repair process.
8. When the repair process is complete, the Install Complete screen appears. Click **Finish**.

---

**Note:** Depending on your operating system, you may need to reboot your system at this point. You will be prompted if a reboot is required; if a message appears, follow the on-screen instructions.

---

## Strategy Design Issues

The following are common issues related to designing your protection strategy. For more information about designing your strategy, see Chapter 4, “Designing Your Protection Strategy,” on page 55.

### Question:

Is there a limit on the number of writes I can make to a cell?

### Answer:

Making frequent writes to a cell can lead to premature key failure. For this reason, Rainbow Technologies recommends limiting writes to a key to a reasonable amount. For example, writing to any cell on a single key once every 10 seconds, 8 hours a day leads to more than 2 million writes a year, which is too many. Write to the key only when necessary; use other means—such as querying or reading a cell—to verify the key is still attached and has not been tampered with.

---

## Application Protection Issues

The following are common issues related to implementing and using application protection as a part of your protection strategy. For a complete description of how to use application protection, see Chapter 7, “Protecting Your Application,” on page 141.

### Question:

Why is one of my files not being encrypted when I apply a shell to my application?

### Answer:

If any of your input files (the application executable, or any other files you have selected for encryption) have the read-only attribute set, SentinelSuperPro may not be able to protect the file.

Clear the read-only attributes in the file’s Properties dialog box, then return to the Implementation stage to apply the shell again.

### Question:

I receive an error if I attempt to encrypt more than 50 files when I apply a shell to my application. How can I encrypt more than 50 files?

### Answer:

When applying a shell, you can encrypt a maximum of 50 files at a time. To encrypt more than 50 files with an application using automatic (shelled) protection, you need to create multiple application protection elements, all of which use the same encryption seed. For example, if you need to encrypt 125 files, you will need to create three application protection elements using automatic protection (50 + 50 + 25).

1. Create an application protection element, and select automatic protection. See “Using Automatic Protection” on page 152.

2. Specify the first 50 files you want to be encrypted at shell time. See “Selecting Additional Files for Encryption” on page 160.
3. Write down the encryption seed you used for the first 50 files.
4. Continue through the Element Definition Wizard to complete the application protection element.
5. Repeat step 1 to add another application protection element.
6. Specify the next 50 files you want to be encrypted at shell time.
7. Enter the encryption seed you wrote down in step 3. **You must use the same encryption seed that you used for the first 50 files, if you want the files to use the same encryption.**
8. Repeat step 4.
9. Repeat steps 5 – 7 until you have specified all the files you want to encrypt.

---

**Note:** *While reading the following sections, keep in mind that even if an application cannot be protected using automatic protection, it can always be protected using integrated protection instead.*

---

### Protecting Multi-File Applications

Using automatic (shelled) protection allows you to protect multiple executable files. However, there are several things you must keep in mind when protecting an application made up of multiple executable (.EXE or .DLL) files:

- You can protect a single DLL that will be used by multiple unprotected .EXE files without any special precautions. However, if you want to protect a single DLL that will be used by multiple *protected* .EXE files, you must make sure the .EXE files and the .DLL file share compatible protection settings. This means they must use the same encryption seed, algorithm data, and data file encryption settings.

- Multiple .EXE files in a single application must have the same encryption seed if they share data files and the same algorithm data (or at least, non-conflicting use of cells).
- If your application uses multiple DLLs, we do not recommend protecting the DLLs, because the last DLL loaded overrides settings for previously loaded ones.

However, if you must protect the DLLs, be very careful to make sure they share exactly the same protection options. The DLLs must also use the same encryption seed and algorithm data as the protected .EXE files in the application.

## Protecting Interpreted-language Applications

A special case of protecting multi-file applications is protecting an interpreted program that uses a separate interpreter program at runtime to execute the application. In this case, you need to protect the interpreter program, as well as your program, since one calls the other.

How you protect the two executable programs depends on the programming system you are using.

In most cases, you will protect the interpreter program and your program as two separate executable files, using the same protection settings: encryption seed and algorithm data.

For instance, to protect a FoxPro 3.0 application, you need to protect the runtime file *vfp300.esl*, as well as your .EXE file.

If your application is an interpreted program that does not require a separate runtime program (such as certain Macromedia applications that append the interpreter to your program in a single executable file), you can protect the single executable file in the usual way.

In some cases, the interpreter program must be protected as the main executable program, and your application must be protected as a data file used by the interpreter.

You may have to experiment to see what works for you. If you need assistance, contact Rainbow Technologies Technical Support.

---

**Note:** *For a list of the compatible compilers and applications supported by SentinelSuperPro, please see “Appendix C, “Compatible Compilers and Applications,” on page 351.*

---

### Input File Attributes

If your input file has the read-only attribute set, SentinelSuperPro may not be able to protect the file. In order for SentinelSuperPro to read the file, you must clear the read-only attributes on the file properties sheet and run the protection application again.

### Thread Local Storage

SentinelSuperPro cannot protect multi-threaded Win32 applications with DLLs that use Thread Local Storage (TLS). Compilers such as Microsoft Visual C++ version 4.2 and higher support TLS.

### Lahey F90 Fortran 2.0

SentinelSuperPro cannot protect a Win32 application that has been compiled with Lahey F90 Fortran 2.0.

### Protecting FoxPro 3.0 and 5.0 Applications

For FoxPro 3.0, both the program (.EXE) and the runtime (*vfp300.esl*) must be protected using the same encryption seed. At least one encrypted data file must be specified; if no real data encryption is needed, specify a dummy filename (for example, *zzz.zzz*). For FoxPro 5.0, only the program (.EXE) and any data files must be protected.

### Protecting Microsoft J++ 1.1 Java Applets

Because Microsoft J++ 1.1 Java uses a separate interpreter program, you must protect both *java.exe* and your applet file as separate executables using the same protection settings.

## Note for SmartHeap Users

If you are using MicroQuill's SmartHeap memory management software and protecting your application as data encryption enabled, you may have difficulty with the encrypted data files under Windows NT. If this is the case, first try to disable the SmartHeap DLL automatic patching activity controlled by Registry values. Please refer to your SmartHeap documentation for more information.

## Protecting Applications That Use "Starter" Programs

Some applications use a "starter program," which in turn calls the main executable application. One application using a starter program is Corel WordPerfect. This application calls various DLLs that call the main WordPerfect application, which in turn may open a document file. In this case, you must protect the main executable file rather than the starter program, and protect the document files as data files used by the main executable application.

## Key Programming Issues

The following are common issues related to programming keys using the SSP Toolkit or the Make Keys Utility. For a complete description of how to program keys, see Chapter 11, “Programming Keys,” on page 213.

### Question:

During key programming, I received a write failure message asking me to enter my developer ID and passwords again. What happened?

### Answer:

If you remove the key while it is being programmed, this message appears.

Verify the key is firmly connected to the appropriate port on the computer, and that it is fully-seated in the port. An inadequate connection can cause these types of errors.

### Question:

A key failed the programming process. How can I tell if the error is software or hardware related?

### Answer:

To determine if a programming failure is due to a software error or a hardware error, try programming another key with the same strategy.

If the programming is successful, the previous error was hardware-related.

If you try programming many keys, and all of them fail programming, the error is software-related.



**Question:**

When I try to create a key prototype in the Prototype stage, or program a key in the Make Keys stage, I get a message saying the server isn't running. What does this mean?

**Answer:**

The SentinelSuperPro Server must be running to access the hardware key. This includes any kind of key access operation, including creating a prototype or programming keys.

For server installation instructions, see “Installing the SentinelSuperPro Server” on page 230. For information about how to verify the server is running, see “Verifying the SentinelSuperPro Server Is Running” on page 261.

---

## Application Activation Issues

The following are common issues related to activating applications or performing field upgrades using the SSP Toolkit or the License Generator Utility. For a complete description of how to activate or update applications, see Chapter 13, “Activating and Updating Keys,” on page 243.

### Question:

Why can't my distributor activate applications or update keys using the License Generator Utility?

### Answer:

There are several possible reasons why a distributor could be having problems activating or updating applications.

- **His distributor key is not connected.** The distributor *must* have his distributor key connected in order to generate license codes for applications using the distributed activation type. Verify that your distributor has his distributor key appropriately and firmly connected.
- **The activation counter on his distributor key has reached zero.** If there are no more licenses to activate or update applications left on the distributor key, your distributor will not be able to generate license codes. You need to increment the activation counter on the distributor's key through field activation. See “Updating Distributor Keys in the Field” on page 257 for instructions.
- **He is using an incorrect distributor key.** The distributor key connected to the workstation must be programmed with the SentinelSuperPro project the application was protected with. If your distributors are responsible for activating multiple applications with multiple keys, it is important that the correct key be connected when attempting to generate license codes.
- **The wrong .DST file is being used.** The .DST file opened in the License Generator Utility must be the file that was created from the

same project you programmed the distributor key with, and protected the application with. If the wrong .DST file is being used, license codes cannot be generated.

- **The SentinelSuperPro Server is not running.** To use the License Generator Utility, the SentinelSuperPro Server must be installed and running on the same workstation the distributor key is connected to and the utility is being run on. Ask your distributor to verify that the server is running, using the instructions found in “Verifying the SentinelSuperPro Server Is Running” on page 261.

# SentinelSuperPro Compatibility

The following table outlines driver and hardware key compatibility between SentinelSuperPro 6.0 and 6.1.

Sentinel Super- Pro Version	Stand-alone	Network	Using SuperPro System Driver		Using SuperPro Hardware Key	
			6.0	6.1	6.0	6.1
6.0	✓		✓		✓	
6.0	✓		✓			update to v. 5.39 of driver
6.0	✓			✓	✓	
6.0	✓			✓		✓
6.1	✓		update to v. 5.39 of driver			
6.1	✓			✓	*	
6.1	✓			✓		✓
6.1		✓	update to v. 5.39 of driver			
6.1		✓		✓	**	
6.1		✓		✓		✓

- ✓ compatible
- \* unlimited licenses without sublicensing
- \*\* only one network license available with no sublicenses

**Note:** Applications that were protected using SentinelSuperPro 6.0 can use 6.1 hardware keys only if version 5.39 of the Sentinel driver is being used. In this case, 6.1 hardware keys are viewed by the protected application as stand-alone keys with a license limit of one.

---

## SentinelSuperPro Key Compatibility Issues

The information in this section will help you isolate and avoid compatibility problems that may occur with the SentinelSuperPro key. This section does not attempt to address compatibility issues with specific models of computers, printer, or other parallel or USB devices.

Rainbow Technologies expends great effort to ensure our products are compatible and transparent to printer operations. Parallel port devices that use the parallel port control lines in Centronics-compatible fashion should also retain compatibility with Rainbow Technologies products.

For the latest information about compatibility issues with specific devices, please contact Rainbow Technologies Technical Support.

### About the Parallel Hardware Interface

---

**Note:** *In the following discussion, all references are to signals on the parallel port connector pins, not to the state of the software registers of the I/O port. Refer to your computer's documentation for more information.*

---

All signal pins, except one on the printer adapter connector, are passed through at all times to the SentinelSuperPro key's outside connector and on to the printer. The SentinelSuperPro key "sits to the side" of these signals, monitoring them.

The BUSY signal return line from the printer is used to send command responses to the PC. The corresponding printer signal is applied to this signal at the parallel port at all times, except when a command is in progress. The SentinelSuperPro key becomes active only when it detects the start of a command.

- **Power** – During operation, the key uses current from all possible signal pins to power its internal logic. Very little power is required to operate the unit.
- **Output** – Data is returned from the key on pin 11 (BUSY).

- **Input** – The key uses the DATA lines on the printer port for input.
- **Reset** – The key is placed in a reset state after a pulse on the STROBE line.

### ***Compatibility Requirements***

The SentinelSuperPro key operates on an IBM-compatible parallel port built to Centronics standards. The key is completely transparent, allowing normal computer-printer communication.

The key connects to the STROBE and BUSY lines, plus two DATA lines. The key uses any of these available lines for power. It receives communications from the computer using the DATA lines, and communicates to the computer using the BUSY line. The key is reset whenever the STROBE line goes inactive.

The SentinelSuperPro key's circuitry inhibits the printer response on the BUSY line from returning to the computer while the key is responding to a command on the corresponding line. This prevents collisions on the response line caused by simultaneous use by the printer and the key. Once the key finishes processing the command, the printer's response is unblocked and allowed to pass through to the computer.

The SentinelSuperPro key meets the electrical parameters for the logic circuits used in the IBM printer adapter card. If you encounter problems using the key with other equipment, compatibility problems may be the cause.

### ***Incompatible Connectors***

Some computers otherwise compatible with the IBM specification include a 36-pin Centronics-type connector on the motherboard, rather than the IBM-style DB-25 connector used with the SentinelSuperPro key.

In this case, you need special printer cables to adapt the SentinelSuperPro key to the computer or the printer.

## ***Using Other Hardware Keys***

If the SentinelSuperPro key is installed on a port with other hardware keys, place the SentinelSuperPro key at the end of the chain (closest to the printer).

## ***Specific Hardware Problems***

Because the SentinelSuperPro key uses the IBM printer adapter as the design model for its hardware interface, it is compatible with the wide range of hardware adhering to the IBM standard for parallel interfaces. However, not all computers properly implement this standard.

For example, some computer parallel ports do not provide pull-up resistors to power the SentinelSuperPro key. This is especially true with some laptop computers. Generally, if a printer is plugged in and powered on, it will provide power.

Because the problems experienced with these systems are caused by use of a non-standard IBM parallel port, they can usually be corrected by replacing the parallel port with a truly IBM-compatible parallel port, or by adding a second, IBM-compatible printer adapter card.

## **About the USB Hardware Interface**

Rainbow Technologies' USB SentinelSuperPro key conforms with the following specifications:

- Microsoft plug-n-play
- Universal Serial Bus

The SentinelSuperPro USB key can be attached to any USB port and has been tested with hub ports, motherboard ports, and some plug-in USB ports.





# Appendix C

---

## Compatible Compilers and Applications

This appendix provides a list of the compatible compilers and applications that can be used to develop applications to be protected with SentinelSuperPro 6.1.

This list applies **only** to applications being protected with the automatic (shelled) protection option.

Applications developed with these and many other languages can always be protected using integrated protection.

## Appendix C – Compatible Compilers and Applications

Language or Application	Product Name	Version	Operating System
Access	Microsoft	97	Win32
Acrobat	Adobe	3.0	Win32
Basic	Microsoft Visual	4.0	Win32
Basic	Microsoft Visual	5.0	Win32
Basic	Microsoft Visual	6.0	Win32
C	Microsoft	6.0	DLL and OBJ
C	Microsoft	7.0	DLL and OBJ
C	Watcom	9.0	DLL and OBJ
C	Watcom	9.5	Win32
C	Watcom	10.0	Win32
C	Watcom	10.5	Win32
C/C++	Watcom	11.0	Win32
C++	Borland	4.5	Win32
C/C++	Borland	5.02	Win32
C/C++	Imprise C++ Builder	3.0	Win32
C++	Microsoft Visual	2.0	Win32
C++	Microsoft Visual	4.1	Win32
C++	Microsoft Visual	4.2	Win32
	Background Check will not work in MFC applications built using MFC's static link library. In order for Background Check to work in these types of applications, you need to install the Visual C++ 4.2B patch. Contact Microsoft, or check their Web site at <a href="http://www.microsoft.com">www.microsoft.com</a> , to obtain the patch.		
C++	Microsoft Visual	5.0	Win32
C++	Microsoft Visual	6.0	Win32
dBase	Borland Visual	5.5	DLL and OBJ
Delphi	Borland	2.0	Win32

## Appendix C – Compatible Compilers and Applications

Language or Application	Product Name	Version	Operating System
Delphi	Borland	3.0	Win32
Delphi	Imprise	4.0	Win32
Director	Macromedia	5.0	Win32
Director	Macromedia	6.0	Win32
Excel	Microsoft	97	Win32
Fortran	Microsoft Powerstation	4.0	Win32
FoxPro	Microsoft	2.5	DLL and OBJ
	Background Check option does not work		
FoxPro	Microsoft Visual	3.0	Win32
	Background Check option does not work.		
FoxPro	Microsoft Visual	5.0	Win32
FoxPro	Microsoft Visual	6.0	Win32
Internet Explorer	Microsoft	4.0	Win32
	With encrypted .HTML files		
Java	Microsoft Visual J++	3.0	Win32
MSPaint	Microsoft	95/98	Win32
	With encrypted .BMP files		
NotePad	Microsoft	95/98	Win32
	With encrypted .HTML files		
Perl	ActiveWare	5.0	Win32
Project	Microsoft	4.1	Win32
Schedule+	Microsoft	7.0a	Win32
WordPerfect	Corel	7.0	Win32
	Shelled wpwin7.exe with .wpd files		



# Appendix D

---

## Glossary

### A

---

#### access code

An attribute that identifies the accessibility and functionality of a cell. Possible values are:

- 0 read/write data word
- 1 read-only (locked) data word
- 2 counter
- 3 algorithm/hidden

*See also* **locked word, hidden word, data word, counter, algorithm.**

#### access mode

Access modes determine where your application will look for the appropriate hardware key. There are three access modes that can be used by your protected application: *stand-alone*, *network* and *dual*. *See also* **network mode, dual mode, stand-alone mode.**

#### action

A group of one or more field activation commands. *See also* **field activation, command.**

<b>activation password</b>	<p>A two-word value that can be used to activate an inactive algorithm. The password is programmed into the two cells immediately following the algorithm. You give your users the password and a utility with which to enter it.</p> <p>Activation passwords use access code 3. They are called hidden words because their values cannot be read by your application. <i>See also</i> <b>access code, algorithm, hidden word</b>.</p>
<b>activation type</b>	<p>Protection provided by SentinelSuperPro that allows for various methods of customer activation of program modules. Possible activation types are <i>active</i>, <i>static</i> and <i>trusted</i>. <i>See also</i> <b>active activation type, static activation type, trusted activation type</b>.</p>
<b>active/inactive bit</b>	<p>A bit in an algorithm's second word that controls whether or not the algorithm will respond correctly to a query. An algorithm must be active to respond to a query. <i>See also</i> <b>algorithm, query</b>.</p>
<b>active activation type</b>	<p>Method of activation provided by SentinelSuperPro where the application is always active. It does not need an activation password (no activation is necessary).</p> <p><i>See also</i> <b>activation type, activation password</b>.</p>
<b>active application</b>	<p>An application that is ready to run when shipped to your customer. It will always remain active, as long as the hardware key is attached.</p>
<b>address</b>	<p>The memory used to identify a specific cell. <i>See also</i> <b>cell</b>.</p>
<b>algorithm</b>	<p>An element containing a bit pattern that defines how the hardware key should encrypt query data sent by your application. The key uses an algorithm to encrypt the query data and then return a value to your application. You design your application to send queries to the key and then evaluate and act upon the responses.</p>

Algorithms can be *active* or *inactive*. Only active algorithms can return a valid response to a query. The *active/inactive bit* in the cell value controls whether or not the algorithm is active.

All algorithms are two words (and thus, two cells) long, and may have activation passwords and counters associated with them.

*See also* **query data, active/inactive bit, query, response value.**

## API

Application Program Interface. The set of client interface routines your application uses to communicate with the Sentinel system driver, which in turn communicates with the hardware key. *See also* **driver.**

## API Explorer

Allows you to experiment with API function calls on various cells in the key before you add them to your source code. It is also a good way to familiarize yourself with the available functions and their uses prior to designing your strategy.

## application protection

An algorithm with an associated activation type as determined by the options you choose to include in your strategy. Application protection can be either *integrated* or *automatic*. The protection type determines when and where software locks are implemented. *See also* **automatic protection, integrated protection, software lock, activation type.**

## automatic protection

Also known as *shelled* protection. The fastest and easiest method of protecting your applications with SentinelSuperPro.

Instead of adding software locks to your source code, a protective “shell” is automatically added to your application’s executable file, so that the software lock is called before the application starts—if the hardware key is not present, the user sees an error message and the application does not run. Automatic protection also gives you more control over demo options such as expiration dates, counters and time/date limits. *See also* **application protection, shell, software lock.**

## C

---

<b>cell</b>	A memory location on the hardware key that holds 16-bit values. Elements occupy one or more cells on the key.
<b>cell type</b>	<p>A code assigned to each cell that defines (logically) how you want to use the cell. The cell type classifies the type of data stored in the cell, which in turn affects how the cell can be used.</p> <p>Each cell type is identified by a two-letter abbreviation; for example, CW identifies a counter word.</p>
<b>cell value</b>	The 16-bit value contained in each cell. The cell value is also known as a <i>word</i> .
<b>Client Activator</b>	<p>An automated license installation utility that is used to create a product-specific activation script for your protected application.</p> <p>The Client Activator is Rainbow Technologies' recommended means of field activation for SentinelSuperPro protected applications, due to its user-friendly interface. The Client Activator also allows your customers to easily and quickly activate your product via a Web site, if you desire.</p>
<b>command</b>	Function calls that describe what will be done to a key in the field. For example, the Decrement Counter command locates the counter cell on the key and decrements it by the value you specify. <i>See also</i> <b>API</b> .
<b>counter</b>	A cell used to count down from a pre-programmed value. The value in a counter is decremented each time your application sends the RNBOsproDecrement() API function. A counter has an access code of 2. Usually, counters are used to control the number of times a demo application is executed. If desired, a counter can be associated with an algorithm; when the counter reaches zero, the algorithm is deactivated automatically. <i>See also</i> <b>access code, demo, algorithm</b> .



## D

---

<b>data word</b>	<p>A cell in a SentinelSuperPro key that is used to store information. A data word can store data such as customer information, serial numbers, passwords, and check digits. You code your application to read the word and then evaluate and act upon the stored value.</p> <p>Your application can use the stored value to verify the key is still attached, or to control program flow or operation. A data word has an access code of 0 (read/write) or 1 (locked/read-only). <i>See also</i> <b>access code</b>, <b>locked word</b>.</p>
<b>decryption</b>	<p>The process of deciphering data that was previously encrypted. Decryption requires a secret key or password. <i>See also</i> <b>encryption</b>.</p>
<b>demo</b>	<p>A demonstration or trial version of an application that uses a counter to control the number of times the application can run before it expires. <i>See also</i> <b>counter</b>.</p>
<b>Design Stage</b>	<p>The Design stage has two sections: Element List View and Element Layout View. Use the Element Definition Wizard, accessible via Element List View, to define cell types and cell values. Element Layout View allows you to view and modify the location of algorithm, counter and data word cells on the hardware key.</p>
<b>developer ID</b>	<p>A unique identification code assigned to you by Rainbow Technologies. You must use your developer ID to program your keys. You must also use it in your protected application to establish a connection with a key.</p>

<b>distributor</b>	Someone outside of your organization who will be responsible for selling and activating your application. For example, distributors could be resellers or fulfillment centers. Distributors must receive a distributor key in order to activate an application using the distributed activation type. <i>See also</i> <b>distributor key, distributed activation type</b> .
<b>distributor key</b>	A key given to your sales distributors, allowing them to perform activation and update functions on product keys provided to end-users when they sell your protected application. A counter decrements each time the distributor activates or updates an application. This allows you to keep track of applications activated by your distributor. <i>See also</i> <b>counter, distributed activation type</b> .
<b>distributed activation type</b>	Method of product activation provided by SentinelSuperPro where the application is inactive until activated by an activation password. The activation password is different for every key; it is derived from the key's serial number, product information, an encryption engine, and an algorithm located on a distributor key. An application using the distributed activation type will be activated by your distributors using a distributor key. <i>See also</i> <b>activation password, distributor key, activation type</b> .
<b>driver</b>	A piece of software that enables the computer to communicate with a peripheral device (the SentinelSuperPro hardware key).
<b>dual mode</b>	An access mode used when you want your application to use either a local key or a network key. This is the default mode for all SentinelSuperPro-protected applications. When in dual mode, an application will send broadcast messages to the network to locate an appropriate server. <i>See also</i> <b>network mode, access modes, stand-alone mode</b> .

## E

---

<b>element</b>	An item in your protection strategy such as an algorithm, counter, data word or application protection.
<b>encryption</b>	The scrambling of data to prevent unauthorized access. Encryption is the most effective way to achieve data security. To read an encrypted file, you must have access to a secret key or password that enables you to decrypt it. <i>See also</i> <b>decryption</b> .
<b>encryption seed</b>	A string of bits used to as an input to an encryption function or algorithm. The larger the seed (the more bits in the seed), the greater the number of potential patterns that can be created, thus making it harder to break the code and decrypt the contents.

## F

---

<b>field activation</b>	<p>A secure method of remotely updating a SentinelSuperPro hardware key's memory after the key is sent to your user.</p> <p>Field activation allows you to increase demo limits, upgrade demo applications to fully licensed versions, and provide access to additional modules or features, without having to ship a new key to the customer or visit the customer's site.</p>
<b>field exchange</b>	<p>Enables you to ship your application in an unusable state, and provide a means for legitimate users to activate it. The activation process is protected by encryption algorithms and passwords pre-programmed into the key. This same process also allows you to support field upgrades and control feature access. <i>See also</i> <b>algorithm, field activation, activation password, active/inactive bit</b>.</p>

## H

---

<b>hard limit</b>	Defines the maximum number of licenses that can be obtained from a key, and thus the maximum number of users (both local and across the network) that can access the protected application. The hard limit is programmed into each key at the factory and cannot be changed. <i>See also</i> <b>hardware key, license, sublicense.</b>
<b>hardware key</b>	The heart of SentinelSuperPro protection. The key controls and verifies access to your protected applications, assuring that only authorized users can run them
<b>hexadecimal</b>	<p>A base-16 number system. That is, a numbering system containing 16 sequential numbers as base units (including 0) before adding a new position for the next number. The hexadecimal numbers are 0-9 and then the letters A-F.</p> <p>When showing the contents of computer storage, one hexadecimal digit can represent the arrangement of four binary digits. Two hexadecimal digits can represent eight binary digits, or a byte.</p>
<b>hidden word</b>	A cell that cannot be read by your application. Most hidden words are algorithms and activation passwords. Your write password and overwrite passwords are also hidden words. A hidden word has an access code of 3. <i>See also</i> <b>access code, algorithm, activation password, write password, overwrite password.</b>

---

**I****Implementation Stage**

This stage allows you to add a shell to an application's executable file, or view the pseudocode protection plan generated during prototyping,

This stage also allows you to define the actions that can be taken through field activation, and is used to create license codes for distribution to customers who have purchased upgrades in the field.

**inactive application**

An application that will not run until it is activated.

**integrated protection**

A form of application protection where software locks (API function calls) are added directly to your source code. It is used to create a custom protection strategy, with control over the amount and location of software locks. *See also* **software lock, API, application protection.**

---

**L****license**

A license allows the user to start the protected application and access the hardware key. Licenses are never physically moved between the server/key and the client workstation. Instead, the SentinelSuperPro server simply keeps track of how many users can run the application and decrements and increments the license count as authorized users are granted permission to run the application and as they exit the application. *See also* **hard limit, sublicense.**

**license code**

A code that describes the actions to be performed on a key in the field. It determines how the application will be activated or updated.

The license code is generated by SentinelSuperPro based on the locking code provided by the customer and the actions you select. When the customer enters the license code in the Client Activator or Field Exchange Utility, a script is automatically run that performs the selected actions on the key.

License codes are unique to the key the locking code was generated from. *See also* **action, command, field exchange, locking code.**

**locked word**

A data word that contains a value that can be read but not changed by your application unless the overwrite passwords are used. A locked data word has an access code of 1. *See also* **data word, access code, overwrite passwords.**

**locking code**

A code that includes information about how a key is currently programmed, including the key's serial number and developer ID. You must have a customer-generated locking code to create a license code. Locking codes are unique for each key. *See also* **action, command, field exchange, license code.**

## M

---

**Make Keys Stage**

The Make Keys stage allows you to program keys prior to distribution. Hardware keys programmed with your protection strategy, as defined in the Design stage, must be distributed with each copy of your software.

**Monitoring Tool**

A Windows application designed for use with protected applications intended to be run on a network. The Monitoring Tool displays information about all SentinelSuperPro servers, keys and user licenses in the field. The tool reports statistics, such as the number of licenses currently in use and the license limit for each key.

## N

---

**network key** Allows multiple network clients to access a protected application using a single hardware key. Network keys, which are typically connected to servers on the network, are programmed at the factory with a hard limit. *See also* **hard limit, hardware key, license**.

**network mode** An access mode used for applications where you want *only* a network key to be used. The application will look for a key *only* on the selected server. If the selected server is not found, or a key is not found on the selected server, the application will *not* send a broadcast message to the network looking for another server and key. *See also* **access modes, dual mode, stand-alone mode**.

## O

---

**Overview Stage** Sections in this stage introduce you to SentinelSuperPro concepts. This stage also features the API Explorer, where you can test API function calls, view the key's cell layout, and send queries to the key to obtain return values.

**overwrite passwords** A set of passwords you must have in order to set or change the value or access code of any cell other than a data word or a cell that is undefined.

Your overwrite passwords are provided to you by Rainbow Technologies. Keep them secure; they have the power to reprogram all unrestricted cells in your key. *See also* **write password**.

## P

---

### product key

See **hardware key**.

### project

A project is stored in a SentinelSuperPro file. The project contains all the data used to create your protection strategy—elements, passwords, your developer ID, algorithm values, counters, data words, field activation commands, etc.

Your project is the template that will be used to program the keys protecting your application.

### Project Stage

This stage provides setup and configuration information. Create or open projects and enter your developer ID and passwords in this stage.

### Prototype Stage

In this stage, you program the cells in the hardware key with the values defined in the Design stage, generating pseudocode for use in adding API functions to your source code. *This stage is a required stage.*

### pseudocode

Outlines the API functions you need to add to your application (if you are using integrated protection), as well as additional information about your protection strategy.

## Q

---

### query

The process by which an application verifies that the hardware key is still attached or has not been tampered with. This is done by sending query data to be scrambled using a specific algorithm stored in the key. See also **algorithm**, **query data**.



**query data**

The value an application sends in a query to the hardware key. The key scrambles the string according to its internal logic and the bit pattern defined in a specified algorithm. It then returns a response to the application. *See also* **response string, algorithm, query.**

**R**

---

**response string**

The scrambled result derived when the hardware key processes query data according to the bit pattern contained in an algorithm. The hardware key returns the response string to the application. The application then uses the response to determine whether the user is authorized to run the application. *See also* **query data, algorithm.**

**S**

---

**server**

The SentinelSuperPro server manages licensing and security for the protected application. The server is the link between the client running your application and the hardware key, located on the network, that responds to the API functions used in your protection strategy.

**shell**

A protective layer wrapped around your application's executable file when you use automatic protection. This layer is encrypted, making it more difficult for a hacker to gain access to your application's code.

All software locks and communication with the hardware key (such as checking and verification) are handled by the shell. An application protected with a shell can be run only if the user has the correct hardware key. *See also* **automatic protection, software lock.**

**software lock**

A decision point in an application. The purpose of a software lock is to verify the presence of the correct hardware key.

For example, an application might send query data to the hardware key, and require a specific response in order to continue execution. Other software locks may simply read the value in a cell and compare it to the value known to be programmed in that cell. *See also* **query data**.

**stand-alone mode**

An access mode used for applications where you want *only* a local key to be used. The application will look for a key *only* on the client machine. If the key is not found, the application will *not* send a broadcast message to the network looking for a server and key. *See also* **network mode, dual mode, access modes**.

**stand-alone key**

A key typically connected directly to a user's local workstation, providing access to the protected application only on a single system. Stand-alone keys have a hard limit of 0, meaning the key can be used only by one user at a time. These keys can also be connected to servers, but provide only a single license at any one time. *See also* **network key, hard limit, license**.

**sublicense**

A sublicense is a license limit you define that is less than or equal to the hard limit programmed into the key. Sublicenses allow you to implement fewer licenses for an application than the hard limit programmed on the key, protect several applications using the same key by defining separate license limits for each, and control concurrent access to specific features or modules within your protected application(s). *See also* **hard limit, license**.

**static activation type**

Method of product activation provided by SentinelSuperPro where the application is inactive until activated with an activation password, unless it is a demo or metered application. The password is the same for every key used with the protected application. *See also* **activation password, demo**.

## T

---

### trusted activation type

Method of product activation provided by SentinelSuperPro where the application is inactive until activated by an activation password, unless it is a demo or metered application. The activation password is different for every key; it is derived from the key's serial number, product information and an encryption engine. *See also* **activation password, demo, activation type**.

## U

---

### USB

Universal Serial Bus. A technology that features one “universal” plug type for all USB peripheral-to-PC connections. USB replaces all the different kinds of serial and parallel port connectors with one standardized plug and port.

USB simplifies the connection of peripherals to computers by providing an instant, no-hassle way to connect USB peripherals. With USB-equipped PCs and peripherals are automatically configured and ready for use.

## W

---

### word

*See* **cell**.

### write password

A password you must have in order to set or change the value or access code of a data word or a cell that is not yet defined. This password also allows you to decrement counters. Your write password is provided by Rainbow Technologies. *See also* **access code, data word, counter**.



# Index

---

## A

- About command 122
- About stage 118
- access codes
  - cells, viewing 127
  - description 36, 37, 355
  - in field activation
    - commands 208
- Access function parameter 124
- access modes
  - default 100, 142
  - definition 355
  - description 100
  - dual mode 100, 101, 104
  - network mode 100, 103
  - setting 100, 102
  - stand-alone mode 100, 103
- access, stepped 89
- accessing
  - online documentation xx
  - printed documentation xxi
  - projects 131
- actions, field activation
  - adding 203
  - default 188
  - description 201–202, 355
  - removing 204
  - selecting 254
  - testing 210
- Activate Algo PW command 207, 209
- activating
  - applications 65, 77, 247
  - keys 244
  - See also* field activation
- activation counter
  - adding update command 202
  - description 61, 66, 209
  - updating 66, 257
- Activation Password (AP) 40
- Activation Password function parameter 124
- activation passwords
  - algorithms 76, 87
  - description 40, 60–61, 356
  - querying 87
  - trusted activation type 151
  - using 76, 90
  - utility for 77, 229
  - with demo applications 92
- activation status 218
- activation types
  - active 60, 356
  - changing algorithm values 149
  - default activation actions 188
  - demo applications 144
  - description 60, 356
  - distributed 61, 64, 323
  - drop-in 323
  - selecting 149, 163
  - static 60, 368
  - trusted 61, 64, 151, 369
- Activation Wizard 151, 249, 250
- active
  - activation type 356
  - algorithms 35, 39, 45, 174
  - applications 59, 60, 356
- Active Algorithm (AA) 39
- active/inactive bit 35, 46, 78, 356
- adding
  - actions 203
  - algorithms 172–176
  - API functions to source code 192–196

## Index

- commands 205
- counters 177–178
- custom elements 172–183
- data words 179–181
- demo counter 147
- field activation
  - actions 203
  - commands 205
- protection to applications 191
- sublicenses 182
- Address function parameter 124
- addresses
  - algorithms 47–51
  - cell 34
  - cell types
    - Activation Password (AP) 41
    - Active Algorithm (AA) 39
    - Algorithm Counter Word (CA) 41
    - Algorithm Half (AH) 40
    - Counter Word (CW) 42
    - Data Word (DW) 43
    - Developer ID (DI) 43
    - Inactive Algorithm (IA) 44
    - Locked Data Word (DL) 43
    - Reserved Word (RW) 44
    - Serial Number (SN) 45
    - Undefined (\*\*) 39
  - description 356
  - selecting 47, 147
- Adobe Acrobat xxii
- advanced
  - encryption techniques 86
  - protection techniques 81–90
- Advanced Editor 4, 324
- ainst.exe 250
- Algorithm Counter Word (CW) 41
- Algorithm Half (AH) 39
- algorithms
  - activation passwords 76, 87
  - active 35, 45, 174
  - adding 172
  - addresses, valid 47–51
  - custom elements 170–176
  - default values 148, 153
  - description 35, 356
  - enhanced engine 45, 47, 174
  - inactive 45–47, 174
  - overriding values 148, 153
  - querying 129
  - using to encrypt data 74–75
  - values 45, 149
  - with counters 50–51, 92
  - with password 49, 51
- API Explorer
  - description 118, 123, 357
  - evaluating functions 194–196
  - invoking functions 124
- API function calls 8
- API functions
  - adding to source code 192–196
  - evaluating 194–196
  - in integrated protection 57
  - invoking 124, 128
  - parameters 124
  - pseudocode 188
  - status codes 315–319
- API, SuperPro 4, 31, 281–314, 357
- APIPACKET 282
- application protection
  - deleting 167
  - description 142, 357
  - editing 166
  - See also* automatic protection
  - See also* integrated protection
  - testing 198
  - testing on network 198
- applications
  - activating 65, 77, 247
  - activation status 218
  - active 59, 60, 356
  - adding protection to 191
  - applying shell to 196
  - calls to key 6
  - compatible with automatic protection 351
  - compiling 211
  - demo
    - activation types 67, 144
    - controlling 41, 90–94
    - description 11, 59
    - protecting 143–144
    - upgrading from 62, 64
  - FoxPro, protecting 340
  - in command-line shell utility 329
  - inactive 59–67, 77
  - interpreted-language 339
  - linking 211

- metered 217, 223, 266
  - multi-file, protecting 338
  - number per key 11, 95–96
  - protecting multiple 166
  - protection methods for 8, 72, 143–144
  - serial numbers, storing 88
  - shipping
    - to customers 228
    - to distributors 238
    - with starter programs 341
  - arrays, condition 89
  - assembly language techniques 88
  - attributes
    - cells 34–37
    - input file 340
  - automatic protection
    - access mode, default 142
    - activation type 61
    - adding shell to executable 196
    - command-line version 327–330
    - compatible compilers 351
    - customizing error messages 165
    - description 9, 58, 142, 357
    - overriding default
      - algorithm values 153
    - selecting
      - cell address 152
      - demo options 156–158
      - executable file 154–156
      - execution control 158
      - expiration date 158
      - files for encryption 160–162
  - input/output files
    - 154–156
    - time control 158
  - troubleshooting 337
  - using 152–166
  - when to use 58, 69
- B**
- Back button 119
  - background checking 159
  - bases, converting between 126–127
  - benefits 10
  - binary values, converting 126–127
  - Bit Mask AND command 207, 209
  - Bit Mask OR command 207, 209
  - Boolean operators 82
  - broadcast messages 101, 104
  - buttons, navigation 119
- C**
- cables, using with keys 28, 216, 222
  - calls, hiding 88
  - cascading keys 29, 215, 221
  - cell types
    - Activation Password (AP) 47
    - Active Algorithm (AA) 39
    - address restrictions 37–38, 45–47
    - Algorithm Counter Word (CA) 41
    - Algorithm Half (AH) 39
    - Counter Word (CW) 42
    - Data Word (DW) 43
    - description 36, 37–38, 358
    - Developer ID (DI) 42
    - groups of 48–51
    - Inactive Algorithm (IA) 44
    - Locked Data Word (DL) 43
    - Reserved Word (RW) 44
    - Serial Number (SN) 45
    - Undefined (\*\*) 38
  - cell values 45, 127, 358
  - cells
    - access codes 36, 127
    - addresses 34
    - attributes 34–37
    - cell types 37–38
    - description 34, 358
    - locations, selecting 147, 152
    - number of 2
    - programmable 10, 34, 36
    - prototyping 188
    - reserved 10, 34–35
    - undefined 38
    - unused 218, 225, 267
    - values 45, 127, 358
    - viewing 127
    - write limit 336
  - changing
    - application protection 166, 189
    - custom elements 184, 189
    - developer ID 133
    - passwords 133, 137
  - Check Algo command 323
  - Check Counter command 323
  - clearing unused cells 218, 225, 267
  - Client Activator
    - customer use of 244, 247, 257

## Index

- deploying 250
  - description 201, 249, 358
  - installing 32
  - requirements 250
  - shipping
    - to customers 228
    - to distributors 238
  - using 151, 249–250
  - client libraries 31
  - closing the Toolkit 139
  - code
    - adding API functions to 192–196
    - examples 194
  - command-line shell utility
    - description 327
    - syntax 328
    - using 329–330
  - commands
    - description 209
    - field activation
      - adding 205
      - default 188
      - description 201, 207, 358
      - randomizing order of 254
      - removing 208
      - testing 210
      - values for 207
    - menu 122
    - SAFE 323
  - compatibility
    - key 347
    - SentinelSuperPro 346
  - compilers, compatible 351
  - compiling applications 211
  - components,
    - SentinelSuperPro 2, 13
  - concurrent access 107
  - conditional access 89
  - connecting keys
    - for distributor programming 221
    - for product key programming 215
  - contact server 100, 103
  - contacting Technical Support xxiii
  - contents, package 17
  - context-sensitive help xx, 123
  - controlling demos 90
  - converting values between bases 126–127
  - Counter Word (CW) 42
  - counters
    - activation 61, 66, 209
    - adding 177
    - custom elements 171
    - decrementing 91
    - description 34, 358
    - moving 186
    - querying 95
    - time or execution control 157
    - using 41–42, 90–95, 143–144
    - values, entering 175
    - with algorithms 50–51, 92
  - counting executions 90
  - creating projects 131
  - custom elements
    - adding
      - algorithms 172–176
      - counters 177–178
      - data words 179–183
    - algorithms
      - adding 172–176
      - description 170
    - valid locations 49
    - with counter 50
    - with password 49
    - with password and counter 51
    - with password and two counters 51
    - with two counters 50
  - counters
    - adding 177–178
    - description 171
  - data words
    - adding 179–184
    - description 171
  - deleting 184
  - description 47, 141, 169
  - editing 184, 189
  - rearranging on key 185–186
  - sublicenses
    - adding 109, 182
    - description 171
  - types 170
  - when to use 170–171
- customers
    - receiving locking code from 252
    - role in field activation 252
    - sending license code to 255
    - shipping applications to 228
    - using Client Activator 244
  - customized
    - error messages 165
    - protection 69



**D**

- DAT files
  - description 324
  - importing 132
- data
  - encrypting with algorithms 74–75
  - garbage 89
  - inserting extra 88
  - stored, reading 72–73
- Data Protection driver
  - files 236
  - installing 236–237
  - shipping
    - to customers 229
    - to distributors 239
    - when to use 154
- Data Word (DW) 43
- data words
  - adding 179–183
  - description 34, 359
  - expiration date 158
  - maximum values 180
  - using 87
  - values in 180
- data, returned 71
- dates, expiration 158
- debuggers, obstructing 90
- decentralizing locks 72
- decimal, converting values 126–127
- Decrement Counter command 207, 209
- Decrement Counter to Zero command 207, 209
- decrementing counters 91
- decrypting files 160, 236
- decryption 359
- default
  - access mode 100, 142
  - actions/commands 188
  - algorithm values 148, 153
  - password values 151
- deleting
  - actions 204
  - application protection
    - elements 167, 188, 189
  - custom elements 184, 189
  - field activation
    - actions 204
    - commands 208
    - project passwords 137
- demo applications
  - activation types 64, 144
  - controlling 41, 90–94
  - description 11, 59, 359
  - licenses for 91
  - limiting executions 93
  - protecting 143
  - upgrading from 63, 65
- demo counter, adding 147
- demo options
  - automatic protection 156–158
  - execution control 158
  - expiration date 158
  - time control 158
- department names 325
- deploying Client Activator 250
- Design stage 118, 119, 359
- Developer Configuration
  - dialog box 113
- developer ID
  - cell type 42
  - changing 133
  - description 112, 359
  - entering 112–113
  - incorrect 113
  - keys 29, 35
- developer product identifier 52
- Developer's Toolkit
  - See* SentinelSuperPro Toolkit
- development languages, compatible 351
- distributed activation type
  - adding update command for 202
  - definition 360
  - demo applications 67
  - description 61, 323
  - example 64
  - selecting 150, 163
- distributor 360
- distributor keys
  - activation counter 61, 66
  - definition 360
  - description 3, 61, 239
  - metering options 224
  - programming 221–225, 266
  - updating 66, 257
- distributors
  - activating applications 65, 247
  - command for updating key 202
  - creating project files for 138
  - responsibilities 247
  - shipping to 238–240
- DLLs
  - encrypting at shell time 160
  - field exchange 113, 114
  - protecting 338

## Index

- documentation
  - online xx
  - printed xxi
  - shipping 240
- drivers
  - Data Protection 154, 236–237
  - definition 360
  - encryption 324
  - installing 24, 229, 236–237
  - packet record 282
  - shipping 229, 239
  - system 6, 229, 236
  - uninstalling 333
  - upgrading 21
- drop-in activation type 323
- DSAFE 323
- dsafe32.dll 114, 115, 239
- dsafedll.dll 239
- DST files 138, 202
- dual mode
  - definition 360
  - description 100
  - finding key in 104
  - setting 101

## E

- editing
  - application protection elements 166, 189
  - custom elements 184, 189
- Element Definition Wizard 118, 145
- Element Layout View 48, 118, 186
- Element List View 118
- elements
  - application protection 166

- custom 169–186
- description 47, 361
- editing 184
- naming 145
- rearranging on key 185
- selecting addresses for 47, 147

- encrypting
  - data 74–75
  - files at shell time 159
- encryption
  - definition 361
  - driver 324
  - seeds
    - entering 162
    - returned values as 82
    - using 85–86
  - selecting files for 160–162
  - techniques 81–86
- encryption seeds
  - entering 162
  - multiplying by 86
  - using 82, 85–86
- encryption techniques 86
- enhanced algorithm engine 45, 47, 174
- entering
  - activation passwords, utility for 77
  - counter values 175
  - developer ID 113–116, 134
  - licence code 278
  - locking code 254
  - passwords
    - key 112
    - project 135
    - values 176
- environment variable 102
- error messages 165

- error-checking 56
- errors
  - licensing 106
  - programming 219, 225
- evaluating
  - API functions 194–196
  - responses 72
- Evaluation Program 4
- EXCLUSIVE OR operator 82
- EXE files
  - protecting multiple 339
  - See also* Automatic Protection
- executable file, selecting 154–156
- execution control 157–158
- executions
  - counting 90
  - limiting 93
- Exit command 122
- exiting the Toolkit 139
- expiration date 158
- Explorer, Internet 17
- exporting to .DST file 138

## F

- features
  - controlling access to 89
  - new 14
  - SentinelSuperPro 10
- field activation
  - actions
    - adding 203
    - default 188, 203
    - description 201, 203
    - removing 204
    - selecting 254
  - commands
    - access codes in 207

- Activate Algo PW 207
  - Bit Mask AND 207
  - Bit Mask OR 207
  - Decrement Counter to
    - Zero 207
  - default 188, 203
  - description 201, 207
  - Increment Cell 207
  - removing 208
  - values for 207
  - Write Cell 207
- customer actions 244
- customer role in 252
- default actions/commands
  - 188, 202
- description 11, 243, 244, 361
- developer actions 244
- testing 210
- testing actions/commands
  - 210
- use of passwords 10
- field exchange
  - description 11, 361
  - See also* field activation
- field exchange DLLs 113, 114, 115
- Field Exchange Utility
  - customer use of 244, 247, 257
  - description 61, 201, 259, 274
  - entering license code 278
  - generating locking code
    - 277
  - installing 274
  - opening 276
  - required files 274
  - shipping 228, 239
  - updating key 278
  - when to use 274
- File menu 122
- files
  - DAT 132, 324
  - Data Protection driver 237
  - decrypting 160, 236
  - dsafe32.dll 114, 239
  - dsafedll.dll 239
  - DST 138, 202
  - encrypting at shell time
    - description 159
    - removing 162
    - selecting 160–162
    - troubleshooting 337
  - executable, selecting 154
  - fieldexchutil.chm 228
  - hhactivex.dll 228
  - hhupd.exe 228
  - HTML Help support 228
  - index.html 21
  - input/output 154–156
  - instdrv.c 237
  - instdrv.exe 237
  - lang\_enu.dll 262
  - licensegen.log 254
  - lockingcode.loc 252
  - log 255
  - makedll.dll 262
  - monitor.exe 228
  - NetSentinel 132
  - readme 16
  - required
    - for Field Exchange Utility
      - 275
    - for License Generator
      - Utility 269
    - for Make Keys Utility 262
  - saving license code to 255
  - sentdata.vxd 237
  - sp\_gXX.dll 262
  - spcommon.dll 262
  - SSP 6\_1 Monitoring
    - Tool.chm 228
  - unsafe32.dll 114, 228, 238, 262
- forgotten passwords 136
- form-factors, keys 27
- FoxPro 340
- functions, API
  - See* API Functions
- G**
- garbage data, using 89
- generating
  - license code 252, 271
  - locking code 277
  - pseudocode protection plan
    - 188
  - query/response pairs 188
  - random values 126, 127
- groups, cell types 48–51
- guidelines
  - for protection 70
  - shipping keys 241
- H**
- handling keys 53, 241
- hard limit 3, 362
- hardware interfaces
  - parallel 347
  - USB 349
- hardware keys
  - activating 244
  - cables for 28, 216, 222
  - cascading 29, 215, 221
  - cell layout 34
  - communicating with 30

## Index

- communication 6, 79
- compatibility issues 347–348
- connecting 28–30, 215, 221
- creating prototype 188
- description 2, 33, 362
- developer ID 29, 35
- distributor 3, 61, 66, 221–225, 239, 257
- drop-in 323
- finding 103
- form-factors 27
- handling 54, 241
- hard limit 3
- installing 27–30, 80
- installing on server 27, 80
- invalid responses 56, 79
- license code 246
- locating 103
- locking codes 245
- master 188
- memory 4, 34
- missing 78–79
- multiple 13
- multiple applications on 11, 95, 96
- NetSentinel 29
- network 3, 27, 215
- on server 27
- ordering 52
- packaging 54, 241
- parallel port 27–29
- passwords 10, 112, 133
- preventing access to 16
- product 3
- programming 36, 95, 213–220, 221–225
- querying frequently 70
- rearranging elements on 185
- response generation 6
- returning 53
- serial number 35, 45
- shipping
  - guidelines 241
  - to customers 228, 239
  - to distributors 238
- stand-alone 3, 27, 53, 215
- testing 220
- troubleshooting 349
- updated by distributors 247
- updating 11, 244, 257, 279
- USB 27, 30
- viewing cells on 127
- hardware requirements 17
- heartbeat messages
  - description 70
  - sending 106
- Help menu 122
- help, context-sensitive xx, 123
- hexadecimal 126–127, 362
- hhactivex.dll 228
- hhupd.exe 228
- hidden word 362
- hiding calls 88
- hiding software locks 81
- Home stage 118
- hotplugging 80
- How? tab 123
- HTML Help
  - requirements 20
  - support files 228, 231
- I**
- identifying network keys 27
- Implementation stage 118, 119, 363
- implementing strategy 187
- importing DAT files 132, 321
- inactive
  - algorithms 45–47, 174
  - application
    - activation example 77
    - description 363
  - applications
    - default actions/commands 202
    - description 59–61
- Inactive Algorithm (IA) 44
- incorrect passwords 113
- Increment Counter command 207, 209
- Increment Distributor Counter command 207, 209
- incrementing activation counter 202, 257
- Index command 122
- index.html 20
- input files
  - attributes 340
  - selecting 154–156
- inserting extra data 88
- installation
  - CD contents 15
  - Data Protection driver 237
  - prerequisites 20
  - repairing 334
  - server 25
- installation Web page 20
- installing
  - Adobe Acrobat xxii
  - Client Activator 32
  - Data Protection driver 236–237
  - Field Exchange Utility 274–275

- hardware key 27–31
- HTML Help support files 231
- interfaces 31
- keys on server 80
- License Generator Utility 269
- Make Keys Utility 262
- Monitoring Tool 231
- SentinelSuperPro 6.1 20
- SentinelSuperPro Server 230, 233
  - on Windows 2000 235
  - on Windows NT 235
  - Windows 9x 233
- system driver 229
- system drivers 24, 229
- instldr.c 236
- instldr.exe 236
- integrated protection
  - activation types 60
  - API functions 57
  - demo counter 147
  - description 8, 57, 142, 363
  - overriding default algorithm values 148
  - selecting 149
  - selecting cell location 147
  - using 146–151
  - when used 57
- interfaces
  - installing 31
  - parallel 347
  - USB 349
- Internet Explorer 17, 20
- interpreted-language applications 339
- invalid responses 56, 79
- invoking API functions 124, 128
- J**
- J++ 340
- Java applets 340
- K**
- Key ID string 245
- keys
  - See* Hardware Keys
- L**
- Lahey F90 Fortran 340
- lang\_enu.dll 262
- language interfaces 31
- languages, compatible 351
- libraries, client 31
- license code
  - copying 255
  - definition 363
  - description 246
  - entering 278
  - generated by distributors 247
  - generating 252, 271
  - one-time update option 115
  - preventing multiple use of 115
  - saving to file 255
  - sending to customers 255
- License Generator Utility
  - description 63, 259, 269
  - generating license code with 271
  - installing 269
  - opening 270
  - required files 269
  - shipping 239
- license ID 106
- license limit 7, 12, 68, 105
- licensegen.log 255
- licenses
  - adding to distributor key 202
  - definition 363
  - description 68
  - enforcement 12
  - errors 106
  - for demos 91
  - hard limit 3, 362
  - heartbeat messages 70, 106
  - maintaining 106
  - obtaining 7, 105
  - releasing 107
  - statistics 5
- limited metering option 224
- limits, cell write 336
- linking applications 211
- loading license code 254
- loadserv.exe 233
- locating keys 103
- Locked Data Word (DL) 43
- locked projects
  - changing passwords 137
  - opening 135
- locked word 364
- locking a project 135–136
- locking code
  - copying 277
  - description 245, 364
  - entering 254
  - generating 277
  - loading from file 254
  - pasting 254
  - receiving 252
  - saving to file 277
- lockingcode.loc 252
- locks
  - See* software locks

## Index

- log file
  - license codes 255
  - server 106

## M

- machine code 87
- maintaining licenses 106
- Make Keys stage 118, 119, 217, 223, 364
- Make Keys Utility
  - description 213, 259
  - installing 262
  - opening 261, 262
  - programming distributor keys 266–268
  - programming product keys 264–265
  - required files 262
  - viewing statistics in 268
- makedll.dll 262
- manipulating returned data 71
- manufacturing code 52
- Manufacturing Utility 4
- master key 188
- meaningless locks 72
- memory used 165
- memory, key 2, 34
- MemView
  - description 127
  - invoking API functions 128
  - using 127–128
  - verifying prototype with 190
- menu commands 122
- merge modules
  - description 229
  - server installation 230
  - using 231
- merging DAT files 133
- messages
  - broadcast 101, 104
  - error 165
  - heartbeat
    - description 70
    - sending 106
- metered applications 217, 223, 266
- metering options, distributor key 224
- methods, protection 72, 143–144
- MicrosoftHTMLHelp.msm 231
- migrating from
  - NetSentinel 325
  - SentinelSuperPro 5.1 322
- missing hardware keys 78–79
- MOD 48
- modes, access
  - See* access modes
- monitor.exe 228
- Monitoring Tool
  - description 5, 364
  - installing 231
- moving
  - counters 186
  - from stage to stage 119
- multi-file applications 338
- multiple
  - applications
    - on one key 96
    - protecting 11, 166
  - DLLs, protecting 339
  - EXE files, protecting 339
  - hardware keys, connecting 29

## N

- names
  - applications 329
  - element 145
  - project, viewing 135
- navigation 117–122
- navigation buttons 119
- navigation pane 119
- NetSentinel
  - keys 29
  - migrating from 325
  - opening files from 132
- network
  - access 12
  - applications
    - demos 91
    - licenses 68
    - protection types 142
  - keys
    - definition 365
    - description 3, 53
    - identifying 27
    - selecting for
      - programming 215
  - licensing 68
  - obtaining license on 7
  - software locks on 7
  - testing protection on 198
- network mode
  - advantages of 101
  - definition 365
  - description 100
  - finding key in 103
  - setting 100
- new
  - features 14
  - project 131
- New command 122
- Next button 119

no-net 100  
 NSP\_HOST variable  
   description 102  
   using 101  
 Numeric Assistant 126–127

## O

obstructing debuggers 90  
 obtaining  
   licenses 7, 105  
   list of servers 104  
   sublicenses 108  
 one-time update option  
   description 115  
   moving element 185  
   results of 256  
 online documentation xx  
 online help xx, 123  
 Open command 122  
 opening  
   Field Exchange Utility 276  
   License Generator Utility  
     270  
   locked projects 135  
   Make Keys Utility 261, 262  
   NetSentinel files 132  
   project 134  
   SentinelSuperPro Toolkit  
     112  
 ordering keys 52  
 orientation pane xx, 122  
 output files  
   overwriting existing 156  
   selecting 154–156  
 overloading 72  
 overriding  
   default algorithm values  
     148

  default password values  
     151  
 Overview stage 118, 119,  
   123, 365  
 overwrite passwords  
   description 10, 112, 365  
   entering 114  
   in field exchange DLLs 115,  
     206  
 overwriting  
   cells on prototype key 188  
   existing encrypted files at  
     shell time 162  
   existing output files 156

## P

package contents 15, 17  
 packaging keys 54, 241  
 packet record 282  
 parallel interface 347  
 parallel port  
   devices 28  
   hardware keys 27–29  
 parameters, API function 124  
 passwords  
   activation  
     algorithms, used as 87  
     querying 87  
     trusted activation type  
       151  
     using 76, 90  
     utility for 77, 229  
     with demo applications  
       91  
   changing  
     key 133  
     project 137  
   description 10  
   entering 135

  key 112  
   project 135  
   values for 176  
 incorrect 114  
 key  
   changing 133  
   entering 112  
   location of 16  
   overwrite 10, 112–114,  
     365  
   project  
     forgotten 136  
     removing 137  
   values 176  
   viewing characters 114  
   with algorithms 49, 51  
   write 10, 112, 369  
 pasting locking code 254  
 PDF files xxii  
 percentages, sublicensing 107  
 product keys 3  
 programmable cells 10, 34, 36  
 programming keys  
   cascading 30, 215, 221  
   connecting keys for 215,  
     221  
   description 36  
   distributor 221, 223, 266  
   errors 219, 225  
   hardware keys 95,  
     214–225  
   manufacturing department  
     role 262  
   overriding activation status  
     218  
   product 215–220  
   removing key during 215,  
     221  
   starting 217, 223

## Index

- statistics 219
  - testing 220
  - troubleshooting 342
  - unused cells 218, 225, 267
  - using Make Keys Utility 264–268
  - Project stage 118, 119, 366
  - projects
    - adding passwords to 135
    - creating new 131
    - description 131, 366
    - for distributors 138
    - importing DAT file 132
    - locking 135–137
    - name 135
    - new 131
    - opening existing 134
    - passwords, forgotten 136
    - providing access to 131
    - saving 135
    - unlocking 137
  - protection
    - adding to application 191
    - application 9, 142–144, 199
    - automatic 9, 152–166
    - customized 69
    - guidelines for 70
    - integrated 8, 142–144, 146–151
    - of multiple applications 11
    - password, for project 135
    - quick 69
    - techniques
      - advanced 81
      - assembly language 88
      - commonly used 72
      - encrypting data 74
      - encryption 81–90
      - hiding calls 88
      - obstructing debuggers 90
      - querying activation passwords 87
      - reading stored data 72
      - returned values as variables 81
      - using activation passwords 76, 90
      - using data words 87
      - using stepped access 89
  - protection plan, pseudocode
    - description 192–193
    - generating 188
  - protection strategy 55, 69
  - protection types
    - automatic
      - description 8, 58
      - using 152–156
      - when to use 69
    - description 8, 57
    - integrated
      - description 8, 57
      - using 146–151
      - when to use 57
    - selecting 145
    - shelled 58
  - prototype
    - creating 189
    - description 188
  - Prototype stage 118, 119, 189, 366
  - prototyping
    - functions 188
    - starting 189
    - verifying programming 190
  - pseudocode
    - description 366
    - generating 188
    - viewing 192
    - with custom elements 169
  - pseudo-random numbers 86
- ## Q
- query
    - API function 307
    - data 6, 367
    - description 366
    - responses to 72
    - sending 70
    - string length 74
    - values 130
  - Query Data function
    - parameter 125
  - Query Response Generator 129–130
  - query/response pairs
    - generating 188
    - in pseudocode 193
  - querying
    - activation passwords 87
    - algorithms 129
    - counters 95
    - hardware keys 71
  - quitting the Toolkit 139
- ## R
- Rainbow Technologies
    - contacting xxiii
    - Web site xxii
  - random
    - query values 130
    - values, generating 126, 127



- randomizing
  - field activation commands 254
  - unused cells 218, 225, 267
- RB\_SPRO\_APIPACKET 282
- Read Cell command 323
- reading stored data 72–73
- readme file 16
- rearranging elements on key 185
- receiving locking code 252
- releasing licenses 13, 107
- removing
  - actions 204
  - application protection elements 166, 188, 189
  - commands 208
  - custom elements 184, 188, 189
  - files to be encrypted at shell time 162
  - project passwords 137
- repairing the install 334
- required files
  - Field Exchange Utility 275
  - License Generator Utility 269
  - Make Keys Utility 262
- required stages 120
- requirements
  - Client Activator 250
  - key compatibility 346, 348
  - system 17
- requirements, hardware 17
- requirements, software 17
- reserved cells 10, 34–35
- Reserved Word (RW) 44
- resetting statistics 219
- response string 367
- response values
  - description 72, 95, 130
  - invalid 56, 79
- restricted cells 10, 34–35
- returned data, manipulating 71
- returned values
  - as encryption seeds 82
  - description 72
  - evaluating 72
  - See also* response values
  - using as variables 81
- returning keys 53
- RMA, obtaining 53
- RNBOSproActivate 76, 78, 286
- RNBOSproDecrement 41, 42, 77, 94, 288
- RNBOSproEnumServer 104, 290
- RNBOSproExtendedRead 292
- RNBOSproFindFirstUnit 78, 293
- RNBOSproFindNextUnit 294
- RNBOSproFormatPacket 126, 295
- RNBOSproGetContactServer 292, 296, 298, 300
- RNBOSproGetFullStatus 297
- RNBOSproGetHardLimit 298
- RNBOSproGetKeyInfo 299
- RNBOSproGetSubLicense 108, 301
- RNBOSproGetVersion 302
- RNBOSproInitialize 126, 304
- RNBOSproOverWrite 305
- RNBOSproQuery 39, 44, 77, 82, 87, 94, 95, 307
- RNBOSproRead 310
- RNBOSproReleaseLicense 107, 311
- RNBOSproSetContactServer 100, 312
- RNBOSproWrite 313
- S**
- SAFE 4, 243
- SAFE commands 323
- SAFE\_CFG 323
- sample code 194
- Save As command 122
- Save command 122
- saving
  - license code to file 255, 277
  - project 135
- scattering code 71
- seeds, encryption 82, 85, 162
- selecting
  - activation type 149, 163
  - demo options 156–159
  - field activation actions 254
  - files for encryption 160–162
  - input/output files 154
  - protection type 145
- sending
  - license code to customer 255
  - queries 70
- sentdata.vxd 236
- Sentinel Client Activator
  - See* Client Activator
- Sentinel Data Protection driver
  - See* Data Protection Driver
- Sentinel driver
  - See* system driver

## Index

- Sentinel Evaluation Program 4
- Sentinel Query Response Generator 4, 129–130
- Sentinel system driver
  - See* system driver
- SentinelShell 4, 323
- SentinelSuperPro
  - API 4, 357
  - benefits 10
  - compatibility 346
  - components 2, 13
  - description 1
  - features 10–14
  - installing 20
  - new features 14
  - package contents 13, 15
  - previous versions 20, 321
  - software protection 6
- SentinelSuperPro 5.1
  - co-existing with 6.0 20
  - migrating from 322, 325
- SentinelSuperPro Advanced Editor 4
- SentinelSuperPro API
  - description 281
  - interfaces 31
  - invoking functions 124, 128
  - status codes 315–319
  - Win-32 281–314
- SentinelSuperPro
  - Manufacturing Utility 4, 323
- SentinelSuperPro Monitoring Tool 5, 231
- SentinelSuperPro SAFE 4
- SentinelSuperPro Toolkit
  - See* Toolkit
- SentinelWizard 4, 132, 324
- Serial Number (SN) 45
- serial numbers
  - application, storing 88
  - for keys 35
  - range of 35, 45
- server, SentinelSuperPro
  - contact 100, 103
  - contacting 104
  - default location 25
  - definition 367
  - description 5
  - executable files 233
  - installing 230
  - installing keys on 27
  - licenses on 105
  - log file 106
  - maintaining license count 106
  - obtaining list of 104
  - responding to broadcast 104
  - system requirements 17
  - updating keys on 278
  - verifying running of 261
- setting access modes 100–102
- setup, starting 20
- shared licensing 325
- shell
  - applying to application 196
  - description 9, 58, 142, 367
  - error messages 165
- shell time file encryption 159
- shell utility, command-line 327–330
- shelling applications
  - See* Automatic Protection
- shipping
  - applications
    - to customers 228
    - to distributors 238–240
  - Client Activator 228, 238
  - documentation 239
  - drivers 239
  - Field Exchange Utility 228
  - hardware keys 239, 241
  - keys 54, 228
  - License Generator Utility 239
  - Make Keys Utility 239
  - Show Passwords check box 114
  - skipping unused cells 218, 225, 267
  - SmartHeap 341
  - SMU 4, 323
  - soft license limits 325
  - software locks
    - decentralizing 72
    - description 6, 8, 368
    - hiding 81
    - in automatic protection 58
    - in integrated protection 57, 142
    - meaningless 72
    - on network 7
    - placement in code 71
    - steps 71
    - variations 10
  - software protection 6
  - software requirements 17
  - software version, viewing 122
  - source code
    - adding API functions to 194–196
    - examples 192
  - sp\_gXX.dll 262
  - spcommon.dll 262
  - spnsrv9x.exe 233

- spnsrvnt.exe 233
- SPROX 324
- SSP 6\_1 Monitoring Tool.chm 228
- stage window 117
- stages
  - About 118
  - description 117
  - Design 118
  - Home 118
  - Implementation 118
  - Make Keys 118
  - moving from 119
  - Overview 118
  - Project 118
  - Prototype 118
  - required 120
- stand-alone
  - access 12
  - applications
    - demos 91
    - licenses 68
    - protection types 142
  - keys 53
    - definition 368
    - description 3
    - selecting for
      - programming 215
- stand-alone mode
  - definition 368
  - description 100
  - finding key in 103
  - setting 100
- stand-alone utilities
  - See* utilities
- starter programs 341
- static activation type 60, 67, 368

- statistics
  - licenses 5
  - programming 219, 268
- status codes, API 315–319
- stepped access 89
- stored data, reading 72–73
- storing serial numbers 88
- strategy
  - creating prototype for 188
  - implementing 187
- sublicense limits 109
- sublicenses
  - adding 109, 182
  - cell type used for 43
  - custom element type 171
  - definition 368
  - obtaining 108
  - value 183
- sublicensing
  - description 12, 68
  - heartbeat messages 108
  - percentages 107
  - usage example 108
  - uses for 107
- subroutines, assembly
  - language 88
- SuperPro 5.1 4, 20, 321
- SuperPro API 4, 31, 281–314
- SuperProMonitor.msm 231
- SuperProNetServers.msm 231
- syntax, command-line shell
  - utility 328
- system driver
  - installing 24, 229
  - shipping 229, 239
  - uninstalling 333
  - upgrading 21
- system requirements 17

## T

- Technical Support, contacting xxiii
- techniques, protection 72–90
- testing
  - application protection 198
  - field activation actions/
    - commands 210
    - programmed keys 220
  - threaded local storage 59, 340
- time control
  - counter values 157
  - description 158
- time-out 13
- Toolkit
  - closing 139
  - navigating in 117
  - opening 112
  - uninstalling 332
- trial versions
  - See* demo applications
- troubleshooting
  - application activation 344
  - cannot update keys in field 114, 115
  - deleted element still visible in MemView 188
  - distributor can't generate license codes 344
  - encrypting more than 50 files at shell time 337
  - file not encrypted at shell time 337
  - hardware problems 349
  - key programming 342
  - repairing installation 334
  - server not running message 343

## Index

- update process unsuccessful 278
- trusted activation type
  - description 61, 67, 369
  - example 62
  - overriding default password values 151
  - querying activation passwords 87

## U

- unauthorized access 16
- Undefined (\*\*) 38
- uninstalling
  - SentinelSuperPro 332
  - system driver 333
- unlimited metering option 224
- unlocking project 137
- untitled.spp 132
- unused cells 218, 225, 267
- Update Key string 246
- update, one-time 115, 185, 256
- updating
  - activation counters 257
  - distributor keys 66, 257
  - keys
    - description 11, 244
    - on server 278
    - randomizing commands 254
    - selecting actions for 254
    - using Field Exchange Utility 278
- upgrading system driver 21
- USAFE 323
- usafe32.dll 114, 115, 228, 238, 274

- USB 349, 369
- USB hardware key 27, 30
- USB ports 30
- user data
  - as query value 130
  - custom elements 171
  - description 10
- utilities
  - activation password 76, 77, 229
  - command-line shell 327–330
  - Field Exchange 274–279
  - installing 24
  - License Generator 269–273
  - Make Keys 262–268
  - previous version 4
  - stand-alone 259–279

## V

- valid addresses 47–51
- Value function parameter 125
- values
  - algorithm 45, 149
  - algorithm, default 148, 153
  - cells 45, 127, 358
  - converting between bases 126–127
  - counter
    - entering 175
    - for time controls 157
  - data words 180
  - field activation commands 207
  - NSP\_HOST variable 102
  - passwords 176
  - query 130
  - random 126–127
  - responses 72, 95, 130

- returned
  - as encryption seeds 82
  - as variables 81
  - sublicense 183
- variable, NSP\_HOST 101, 102
- variables, returned values as 81
- verifying
  - prototype 190
  - server is running 261
- viewing
  - cells on key 127
  - password characters 114
  - programming statistics 219, 268
  - project name 135
  - pseudocode 192
  - software version 122

## W

- Web site
  - installation 20
  - Rainbow Technologies xxii
  - Technical Support xxiii
- What? tab 123
- What's This? help xx, 123
- Why? tab 123
- Windows 17
- Windows 2000
  - installing server on 235
  - verifying server is running on 261
- Windows 9x
  - installing server on 233
  - verifying server is running on 261
- Windows Installer
  - description 21, 229
  - using merge modules 231

- Windows NT
  - installing server on 235
  - verifying server is running
    - on 261
- words
  - algorithms 35
  - counter 34
  - data 34, 87
  - description 34, 45, 369
  - types 34
- Write Cell command 207, 209
- write password
  - description 10, 112, 369
  - entering 114
- writing to cells, limitations 336

## **X**

- XOR operator 82, 85, 86

