



***BATTLE-PROVEN
SOFTWARE
PROTECTION
AND
LICENSE SECURITY***

SDK User Manual

BATTLE-PROVEN SOFTWARE PROTECTION & LICENSE SECURITY
SINCE 1992

CrypKey SDK User Manual

RELEASE DATE: JANUARY 2003

DOCUMENT VERSION: 6.003

© 2003 CrypKey (Canada) Inc.
Calgary, Alberta, Canada

Phone 1-403-398-8011 • *Fax* 1-403-264-8838

Sales email sales@crypkey.com

Support email support@crypkey.com

<http://crypkey.com>

CrypKey License Agreement

Grant Of Rights

In consideration of payment, CrypKey (Canada) Inc. ("CrypKey") grants to the purchasing company (the "Customer") the non-exclusive right to possess, use, and make and distribute unlimited copies of the machine-executable code version of the CrypKey Software Licensing System, including all revisions, modifications, and updates thereto furnished to the customer, together with the written User Manuals in relation thereto (collectively the "Software").

Intended Purpose

The Software is intended to be linked with and incorporated into the product, including, but limited to, the foundation module and installation routines forming a part thereof. No right to distribute the Software on a stand-alone basis is intended by this License Agreement. Any other use of the Software requires written consent from CrypKey. The customer may not reverse engineer, modify, nor create derivative works based on the Software without written consent from CrypKey, except as permitted by acceptable law.

Ownership

All patents, copyrights, and other proprietary rights in the Software are, and shall remain, the exclusive property of CrypKey or its suppliers. The Customer may not assign nor transfer the rights granted in the License Agreement to any person, except the Customer's subsidiaries and affiliates, without written consent from CrypKey.

Limited Warranty

CrypKey warrants to the Customer that the Software will operate essentially as described in CrypKey brochures and documentation. CrypKey warrants that the media on which the Software is recorded shall be free from defects in materials and workmanship under normal use and service for 60 days from the date of the Customer's invoice. If failure of the media is a result of accident, abuse, or misapplication of the Software, CrypKey shall not be responsible for its replacement. Applicable law may imply warranties that cannot be excluded or can be excluded only to a limited extent. This Agreement shall be reached and construed subject to such laws.

No Other Warranties

THE LIMITED WARRANTY SET FORTH HEREIN IS IN LIEU OF, AND CRYPKEY DISCLAIMS, ANY AND ALL OTHER WARRANTIES (EXPRESS OR IMPLIED) WITH RESPECT TO THE SOFTWARE, INCLUDING ANY AND IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Remedies

Except in connection with a claim for infringement of copyright, patent, or other intellectual property right, the Customer's sole and exclusive remedy for a breach of warranty shall be: (a) the return of the initial fee paid for the rights granted herein or (b) the correction or replacement of defective software or media. Corrected or replaced Software or media will be warranted to the same extent as the original Software or media for the remainder of the original warranty period or 30 days from the date of receipt by the customer, whichever is longer.

Limitation Of Liability

IN NO EVENT SHALL CRYPKEY BE LIABLE TO THE CUSTOMER FOR INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGE. IN NO EVENT SHALL CRYPKEY CANADA'S AGGREGATE LIABILITY TO THE CUSTOMER EXCEED THE AMOUNTS PAID TO CRYPKEY BY THE CUSTOMER FOR THE SOFTWARE.

Copyright ©2003, CrypKey (Canada) Inc. All rights reserved

The software described in this manual is furnished under a License Agreement and may only be used in accordance with the terms of this agreement.

All trademarks are the property of their respective owners.

Microsoft, Windows, Visual Basic, Access, Windows NT, Windows 95, and Windows 98 are trademarks of Microsoft Corporation. Watcom is a trademark of Watcom International Corporation. PowerBuilder is a registered trademark of Powersoft Corporation. Novell is a registered trademark of Novell, Inc

Table of Contents

SECTION	PAGE
1 Introduction	1
1.1 About CrypKey (Canada) Inc.	2
1.2 About CrypKey SDK	2
1.3 How to Use This Book	3
1.3.1 Summary of Contents	4
1.3.2 Strategies for Using This Manual	5
1.3.3 Special Icons	6
1.4 System Requirements	6
1.5 Technical Support	7
1.6 About Your CrypKey License	8
1.6.1 Registering with CrypKey (Canada) Inc. to Get Your Developer Keys	9
1.6.2 Using the Temporary License for Development	9
1.6.3 Obtaining Authorization for Your Site Key Generator	10
1.7 CrypKey Licensing Implementation	13
1.8 New Features in Version 6.0	15
1.8.1 CloneBuster™ Technology	16
1.8.2 EasyLicense	17
1.8.3 Dynamic Encryption	17
1.8.4 Enhanced Anti-Hacking	17
1.9 Updating to CrypKey SDK v6.0	18
2 Installation	21
2.1 Getting Started	21
2.1.1 Installing CrypKey SDK from CD	22
2.1.2 Downloading and Installing CrypKey SDK	32
2.1.3 Packing List and Readme Files	38

2.2	CrypKey License Service Installation	40
2.3	Windows 9x/ME Network Installation	41
2.4	MS-DOS Network Driver Installation	42
2.4.1	Example	43
2.5	Novell NetWare Installation	43
2.6	Libraries and Distribution Files	44
2.6.1	16-Bit Libraries	44
2.6.2	WIN32 Libraries	44
2.6.3	32-bit Thunk Library Installation	46
2.7	Connectivity to Microsoft Access	47
3	Demonstration Programs	49
3.1	Authorizing a Windows Program	49
3.2	Authorizing an MS-DOS Program	55
4	Site Key Generator	57
4.1	Levels and Options	57
4.1.1	Levels	58
4.1.2	Options	58
4.2	Additional Features of the Site Key Generator	59
4.2.1	License Duration Settings	59
4.2.2	EasyLicense	60
4.2.3	CloneBuster™ Technology	61
4.2.4	Transaction Summary	62
4.3	Using the Site Key Generator	63
4.3.1	Generating Site Keys	64
4.3.2	Configuring Your Products	68
4.3.3	Incrementing Existing Licenses	75
4.4	Distributor Authorizing License	77
4.4.1	Authorizing Distributors	77
5	Programming Your Application	85
5.1	Basic Programming Steps	85
5.2	GUI Programming Tasks	88
5.2.1	Authorizing the Program	89
5.2.2	Registering the Authorization Transfer on the Target Computer	89
5.2.3	Transferring the Authorization Out of the Source Computer	90
5.2.4	Transferring the Authorization into the Target Computer	90

5.3	Compiling and Linking	90
5.3.1	CrypKey Header File	90
5.3.2	Linking	91
5.4	COM Object	94
5.4.1	COM Object Identifiers	95
5.4.2	Installing a COM Object	95
5.4.3	Referencing a COM Object Explicitly	96
5.4.4	Referencing a COM Object by Name	97
5.4.5	Pre-defining DLL functions	97
5.4.6	Using a COM object in C++	98
5.4.7	COM Object Constants	100
5.4.8	Internal COM Status Messages	101
5.4.9	Referencing the Floating License Snapshot Record	102
5.5	External Linking using Binary Files	104
5.5.1	External Linking Security	104
5.5.2	Transferred Information	105
5.5.3	External Text Link	107
5.6	Creating a Windows NT License Service Installation	107
5.6.1	Error Codes	108
5.6.2	Run Mode	109
5.6.3	Installation Strategies	109
5.6.4	Uninstalling the Driver	110
5.6.5	Testing	111
5.6.6	Supported Versions of Windows NT	111
6	Protecting Your Software with StealthPLUS™	113
6.1	Visual C++ Custom Step Example:	115
6.2	StealthPLUS™ File Auto-Distribution	115
6.3	Using the Interface (STELTHUI)	116
6.4	Using the Command Line Version (STELTHCM)	120
7	Counter-Hacker Strategies	121
7.1	Anti-tampering Measures	121
7.1.1	Error Codes	123
8	Network Licensing	125
8.1	Implementing Floating Licenses	125
8.1.1	Providing Multi-User Functionality	126

8.1.2	Installing a Network License Server	126
8.1.3	Specifying a Floating License Type	126
8.2	CrypKey Network Drivers	127
8.2.1	Supported Platforms	127
8.2.2	Installation Procedures	128
9	Moving Protected Programs	131
9.1	Direct License Transfers	131
9.1.1	Example	132
9.2	Floppy Disk License Transfers	132
10	Frequently Asked Questions	135
10.1	General	135
10.1.1	Software Protection	135
10.1.2	CrypKey Advantages	136
10.1.3	Telephone Authorization	138
10.1.4	Client Annoyance	138
10.1.5	24-hour Telephone Support	139
10.1.6	Computer Signatures	139
10.1.7	License Transfer to Floppy Disk	139
10.2	Technical	140
10.2.1	Disk Compression Programs	140
10.2.2	Watcom Support	140
10.2.3	Library Impacts	140
10.2.4	Hard Disk Failure	140
10.2.5	Product Update Releases	141
10.2.6	Multiple Program Names	141
10.2.7	Floating Licenses	141
10.2.8	Computer Clock Manipulation	142
10.2.9	CD-ROM Distribution	142
10.2.10	Internet Distribution	143
10.2.11	File Set Protection	143
10.2.12	Foreign System Compatibility	143
10.2.13	Removable Hard Drives	143
10.2.14	Anti-Virus Software	144
10.2.15	Installer Software	144
10.2.16	US Exports	144
10.2.17	Software Certification	144
10.2.18	Windows 98	144

11	Troubleshooting	145
11.1	CrypKey Error Messages	145
11.1.1	General Errors	145
11.1.2	Function Errors	146
11.2	General Error 102	146
11.2.1	CrypKey License Service Problems	147
11.2.2	CrypKey Technical Support	150
11.3	Windows NT Diagnostics	151
11.4	NetWare-related Issues	151
11.4.1	Sample Error 1	152
11.4.2	Sample Error 2	152
11.4.3	Sample Error 3	152
11.5	Norton Utilities Speed Disk	153
12	Function Reference	155
12.1	Function Summary	155
12.2	General Error Values	158
12.3	Function Descriptions	162
12.3.1	AcquireLicense()	162
12.3.2	CKChallenge()	165
12.3.3	CKChallenge32()	167
12.3.4	CKTimeString()	169
12.3.5	CrypkeyVersion()	170
12.3.6	DirectTransfer()	171
12.3.7	EndCrypkey()	174
12.3.8	ExplainErr()	175
12.3.9	ExplainErr2()	176
12.3.10	FloatingLicenseGetRecord	178
12.3.11	FloatingLicenseSnapshot	179
12.3.12	Get1RestInfo()	181
12.3.13	GetAuthorization()	183
12.3.14	GetAuthorization2()	191
12.3.15	GetDrivePermanentSerialData()	192
12.3.16	GetLastError()	192
12.3.17	GetLastErrorCode()	192
12.3.18	GetLevel()	193
12.3.19	GetNetHandle()	194
12.3.20	GetNumCopies()	195

12.3.21	GetNumMultiUsers()	195
12.3.22	GetOption()	196
12.3.23	GetRestrictionInfo()	197
12.3.24	GetSiteCode()	200
12.3.25	GetSiteCode2()	201
12.3.26	InitCrypkey()	202
12.3.27	KillLicense()	209
12.3.28	ReadyToTry()	211
12.3.29	ReadyToTryDays()	218
12.3.30	ReadyToTryRuns()	219
12.3.31	RegisterTransfer()	221
12.3.32	SaveSiteKey()	223
12.3.33	SetNetHandle()	225
12.3.34	TransferIn()	226
12.3.35	TransferOut()	231
13	Appendix I – Sample File	236
13.1	Example.c	237
14	Appendix II – Quick Reference	245
14.1	CrypKey Functions	245
14.2	CrypKey Return Codes	248
14.3	CrypKey 6.X OS File Distribution Matrix	249
15	Glossary	253
16	Index	257

List of Tables and Figures

TABLE	PAGE
Table 1: Files required for distribution with 32-bit Applications	44
Table 2: Effects of Duration Settings for Slave Site Key Generator	78
Table 3: Libraries	91
Table 4: Library Linkages	93
Table 5: COM Object Identifiers	95
Table 6: Internal COM Status Messages	101
Table 7: COM C++ and C#.NET Prototypes for FLS Record Data Items	102
Table 8: Transferred Information Descriptions	105
Table 9: Software Protection Technology – Problems and Solutions	136
Table 10: Function Summary	156
Table 11: General Errors and Solutions	159
Table 12: DirectTransfer() Values	172
Table 13: Functioncode Values	176
Table 14: Functioncode Values	177
Table 15: Floating License Snapshot – Return Values	180
Table 16: Floating License Snapshot – Record Structure	181
Table 17: Get1RestInfo() Values	182
Table 18: GetAuthorization() Values	184
Table 19: GetNumCopies() Values	195
Table 20: GetOption() Values	197
Table 21: Authopt Values	198
Table 22: Num_used Values	198

Table 23: GetSiteCode() Values	201
Table 24: InitCrypkey() Values	205
Table 25: KillLicense() Values	210
Table 26: ReadyToTry() Values	213
Table 27: RegisterTransfer() Values	222
Table 28: SaveSiteKey() Values	224
Table 29: TransferIn() Values	228
Table 30: TransferOut() Values	233

FIGURE	PAGE
Figure 1: Site Key Generator Startup	11
Figure 2: License Window – Not Authorized	11
Figure 3: License Window – Site Key Inserted	12
Figure 4: License Window – Authorized	13
Figure 5: CrypKey SDK Program Group	22
Figure 6: Opening CrypKey CD Installation Window	23
Figure 7: CD Installation Screen 1	24
Figure 8: CD Installation Screen 2	25
Figure 9: CD Installation Screen 3	26
Figure 10: CD Installation Screen 4	27
Figure 11: CD Installation Screen 5	28
Figure 12: CD Installation Screen 6	29
Figure 13: CD Installation Screen 7	30
Figure 14: CD Installation Screen 8	31
Figure 15: Download Installation Screen 1	32
Figure 16: Download Installation Screen 2	33
Figure 17: Download Installation Screen 3	34
Figure 18: Download Installation Screen 4	35
Figure 19: Download Installation Screen 5	36
Figure 20: Download Installation Screen 6	37
Figure 21: Download Installation Screen 7	38
Figure 22: Program Group	49

Figure 23: Sample Programs Group	50
Figure 24: Opening WEXAMPLE.EXE (TESTAPP) Window	51
Figure 25: TESTAPP Window with Site Code	52
Figure 26: Site Key Generator Window with Validated Site Code	53
Figure 27: Generated Site Key	54
Figure 28: TESTAPPS Window with Authorized Site Key	55
Figure 29: CrypKey SDK v6.0 Program Group Popup Window	56
Figure 30: Site Key Generator – License Generation Summary Window	62
Figure 31: CrypKey Site Key Generator 6.0 Window	63
Figure 32: Site Key Generator – Starting Authorization	65
Figure 33: Hard Drive Serial Number Information Window	66
Figure 34: HDSN License Details Window	67
Figure 35: Configure Window	69
Figure 36: Site Key Generator with New Application Popup Window	70
Figure 37: Configuration Window with New Application	71
Figure 38: Site Key Generator – Options Tab Page	72
Figure 39: Site Key Generator – Options Combo Box	73
Figure 40: Site Key Generator – Levels Tab Page	74
Figure 41: Site Key Generator – Levels Combo Box	75
Figure 42: CrypKey Site Key Generator 6.0	76
Figure 43: Distributor's Unauthorized SKG License Window with Site Code	80
Figure 44: Master Site Key Generator with Distributor's Site Code	81
Figure 45: Master Site Key Generator with Slave Site Key and Selected Options	82
Figure 46: Site Key Generator License Authorization	83
Figure 47: Slave SKG License Window – Authorized	84

Figure 48: StealthPLUS™ Opening Window	116
Figure 49: StealthPLUS™ Validated Window	117
Figure 50: StealthPLUS™ Configuration Window	118
Figure 51: StealthPLUS™ Compression Begun	119
Figure 52: Network Disconnect Message Window	147
Figure 53: NT File System Troubleshooting Flowchart	148
Figure 54: Novell File System Troubleshooting Flowchart	149
Figure 55: .ckn Diagnostic File	151
Figure 56: Options and Levels	190
Figure 57: CrypKey Program Flowchart	250
Figure 58: Example Win9X/ME CrypKey Configuration	251

Introduction

Welcome to CrypKey SDK and the CrypKey world of software protection.

CrypKey is a revolutionary Software Licensing Application Program Interface (SLAPI) that provides the serious software development company with invisible security and complete flexibility over the licensing of its products. The CrypKey system is designed to replace any external hardware (dongle) or floppy disk-based key system in copy protection effectiveness.

With the CrypKey Software Developer's Kit (SDK), you (the software developer) can:

- protect a software product from unauthorized use on a turnkey basis
- grant customers different levels of privileges
- grant customers usage privileges limited by time or number of uses

You can provide all of these services via telephone, fax, or email.

CrypKey SDK:

- uses no hardware key or disk key
- includes a software library that easily integrates with the developer's existing programs

- can be used to track and control protected software sold by third party distributors
- defeats hard disk drive (HD) cloning of your software

1.1 About CrypKey (Canada) Inc.

The unique CrypKey copy protection and license control software came into being in 1992 with the goal of developing a better way for a large engineering company to protect its own software. CrypKey worked so well, that it soon became its own commercial product and quickly outsold the product it was created to protect!

The CrypKey product and service business was spun off by current management into CrypKey (Canada) Inc. in March 2002. The CrypKey team has developed new, ultra-secure, industry-first CloneBuster™ technology for its version 6.0.

CrypKey has delivered battle-proven software copy and license protection for thousands of companies worldwide since 1992, including: Nokia, Ericson, IBM, Microsoft Caterpillar, Eaton, Dupont, Fujitsu, Minolta, HP, Honeywell, 3M, Lockheed Martin, Mobil, Procter&Gamble, Siemens, Verizon, Sybase, Bell&Howell, Kodak, Allen Bradley, and ABB.

1.2 About CrypKey SDK

CrypKey SDK includes a trial period that allows you to evaluate the full functionality of the system. Obtain your license only when you are satisfied with CrypKey SDK's performance.

This user manual contains example User and Master Keys (see below) for use during your trial period. These temporary keys are **not** intended for distribution with your product. If you distribute your product with these temporary User and Master Keys, recipients who have CrypKey can unlock your product. Since all Site Key Generators can create keys for any program that uses the temporary keys, the only keys you should distribute with your product are those specifically created for the product.

Example Keys

- User Key: D050 815C D1A2 A79D B103
- Master Key (16-bit): 2A5D 57C4 1B4C 135B F09E 17F7 600B 2D70 79E8 F275 C36A
- Master Key (32-bit):

F2C938D2D34678D2E9217C18D78EA6A8E466CF49520F92CDD1B6916\BD
460D60E7C7B4CC7CC1750BD7188F90AC132B915E82FC8FA60A1D299D
A0F28EA3C66BD42DB\0BE62149DAAEE4DBA55C0E70CE1C13BD343F8
B7573ABC1E7DA0695955AB2BD377F50A9BE29A\04CF816B30CD171E15
09AD65100C999E52A35F45A215212A970

CrypKey SDK offers two types of developer keys—those related to the example file and those locking CrypKey to your product. Think of keys as alphanumeric strings of text. You require the following four keys to run CrypKey SDK:

- User Key: created from the CrypKey password you specify
- Master Key: created from the file name you give to CrypKey and other pertinent information
- Site Code (generated from your computer on which CrypKey SDK is installed): required in order to obtain your site key
- Site Key (for the Site Key Generator): enables you to authorize your product using your Site Key Generator

1.3 How to Use This Book

This manual is about two business relationships:

- your relationship, as a software vendor, with CrypKey Canada Inc.: how you obtain and utilize the CrypKey Software Developer Kit to protect your software as you distribute it to customers
- your relationship with your customers and distributors: how you provide, protect, and license your applications

Please note that CrypKey SDK is designed for software distributors. All material in this manual assumes competence on your part in the use of Windows-based programs.

Following is a summary of the manual's contents, as well as strategies for using it.

1.3.1 Summary of Contents

Chapter 1, *Introduction*, contains the following information:

- system requirements
- technical support available at CrypKey Canada Inc.
- getting authorization to use CrypKey to license your applications
- general CrypKey licensing process
- new features in CrypKey 6.0
- procedures for updating to CrypKey 6.0

Chapter 2, *Installation*, focuses on the installation and setup of the CrypKey software and associated drivers (Windows NT/9x/ME/2000/XP, MS-DOS), files, and software.

Chapter 3, *Demo Programs*, provides two examples of CrypKey operation—one for the Windows environment and the other for the MS-DOS environment.

Chapter 4, *Site Key Generator*, discusses the core of the CrypKey interface, namely the Site Key Generator, which is used for configuring the protection features for your software and issuing licenses to your customers and distributors.

Chapter 5, *Programming Your Application*, discusses the basic steps in programming CrypKey protection features and license information into your application. It discusses the CrypKey libraries and how to link to them. It also includes instructions for using the CrypKey COM Objects, which are tools that facilitate and simplify the use of libraries.

Chapter 6, *Protecting Your Software with StealthPLUS™*, discusses the use of CrypKey's StealthPLUS™ product to protect your distributed application against hackers. Supplementary information on this issue is provided in Chapter 7, *Counter-Hacker Strategies*.

Chapters 8, *Network Licensing*, and 9, *Moving Protected Programs*, deal with network licensing and license transfers by customers.

Chapter 10, *Frequently Asked Questions*, discusses general and technical questions that users commonly raise about CrypKey SDK.

Chapter 11, *Troubleshooting*, provides strategies for identifying problems and dealing with the most common CrypKey errors, the most frequent error being Error 102 (produced by the `InitCrypKey()` function). This chapter also deals with CrypKey License Service errors, Windows NT diagnostics, NetWare issues. As well, it identifies the general information that you need to provide to CrypKey technical support staff if you have a problem requiring their assistance.

Chapter 12, *Function Reference*, provides detailed information, including usage, syntax, parameters, and errors, concerning all programming functions used in CrypKey. This chapter includes a summary of function descriptions. For a quick reference of function syntax, see Appendix II.

Chapter 13, *Appendix I*, contains the code for a sample CrypKey executable, demonstrating the use of CrypKey library functions.

Chapter 14, *Appendix II*, contains quick-reference information about functions and errors.

At the end of the manual are a *Glossary* and *Index*.

1.3.2 Strategies for Using This Manual

If you are new to CrypKey, we advise you to read carefully the *Introduction*, *Installation*, and *Demo Programs* chapters and follow the instructions to try out the CrypKey functionality on your computer. After reviewing these chapters and examples, you'll be ready to work with the main procedural chapters of the manual—Chapter 4, *Site Key Generator*, and Chapter 5, *Programming Your Application*.

The *Site Key Generator*, *Programming Your Application*, and *Function Reference* chapters are your main resources for day-to-day use of CrypKey SDK. In addition, *Appendix II* is useful for quick-reference information about functions and errors. Specifically, Sec. 14.2: CrypKey Return Codes discusses errors returned by the two CrypKey functions that **must** be called for all operations—**InitCrypkey()** (see Sec. 12.3.26) and **GetAuthorization()** (see Sec. 12.3.13).

If you are experiencing a problem with CrypKey, we suggest initially reviewing the *Frequently Asked Questions* and *Troubleshooting* chapters, as well as *Appendix II—Quick Reference*. If the problem identification and possible solution do not come to light from these sections, look for more detail in the *Function Reference* and *Programming Your Application* chapters. If it is not readily apparent how to resolve the situation, then please contact CrypKey technical support—see Sec. 1.5: Technical Support.

1.3.3 Special Icons

The following icons are used throughout the manual to convey the meanings shown:



DON'T FORGET



GLAD COMPUTER



WATCH OUT

1.4 System Requirements

CrypKey is designed for use by software developers. In this manual, we assume that you have a good working knowledge of your programming environment, but we don't necessarily expect you to be an expert programmer.

CrypKey SDK requires the following minimum PC hardware requirements:

- a 486 (or better) processor with 64 MB of RAM and 25 MB of free space on your hard drive
- a VGA color monitor

- a mouse
- Windows 3.1, Windows 95, Windows 98, or Windows NT platforms (NT, 2000, XP)

CrypKey SDK supports the following development platforms:

- Microsoft C++ 6.0, 7.0, and 8.0 (directly, using static libraries)
- Microsoft Visual C++ 1.5, 2.x, 4.x, 5.x and 6.x, and C++.Net (directly, using static libraries)
- Borland C++ 4.x (directly, using static libraries)
- Borland C/C++ 3.x or more recent
- any development system capable of reading and writing a file via external linking (for MS-DOS) or DLL (for 16- or 32-bit Windows). This includes, but is not limited to, Microsoft Visual Basic, Microsoft FoxPro, Microsoft Access, CA Clipper, Borland Delphi, Borland Visual dBASE, Borland Paradox, and Powersoft PowerBuilder.
- .Net — VB.Net, C#.Net, C++.Net

1.5 Technical Support

CrypKey SDK includes 60 days of technical support, starting from your date of purchase. Our customer service representatives, while always pleased to take your call, ask that you review the following checklist before requesting their assistance:

- read the manual (most problems can be solved using the information contained in this document)
- work through the demonstrations in Chapter 3, which provide a basic understanding of CrypKey SDK
- test the sample program within the scenarios you are investigating (again, most questions can be answered using this method; if problems persist, try the same situation with example.exe)

If these alternatives do not satisfy your requirements, please contact us by phone or email. Our business contact information is listed below.

CrypKey (Canada) Inc.*World Headquarters:*

Mailing Address	CrypKey (Canada) Inc. The Devenish Heritage Building 908 - 17th Avenue SW Suite 208 Calgary, Alberta T2T 0A3 Canada
-----------------	--

World HQ Phone:	1-403-398-8011
Regular Business Line:	1-403-258-6274
Fax Line:	1-403-264-8838
Support email	support@crypkey.com
Internet	http://crypkey.com

Pacific Rim Office:

Email	Greg.Wible@crypkey.com
Phone:	1-562-989-8976
eFax:	1-208-978-8503

1.6 About Your CrypKey License

By design, the CrypKey system allows CrypKey to send you this developer's kit now and receive payment later, after you are completely confident in CrypKey's abilities. However, if you want to ship CrypKey with your product, you need the following developer keys and numbers:

- **User Key:** This is an encrypted form of the developer/product-specific password and is provided to you by CrypKey. Use this key for the **InitCrypkey()** function.
- **Master Key:** This is an encrypted form of information specific to the product being protected and is provided to you by CrypKey. Use this key for the **InitCrypkey()** function.
- **Site Key:** This key is required to unlock any CrypKey-protected software. Since the Site Key Generator is protected, you must send us the site code so that we can authorize your Site Key Generator.
- **Company Number:** This value is specific to your company and is used for the **CKChallenge()** function.

- **Password Number:** This value is specific to your password and is used for the **CKChallenge()** function.

Although you can develop your application without the developer keys using the temporary license provided, you eventually need the keys for final testing and shipping of your product. These keys are issued only after we receive your payment. If you are working under an aggressive development schedule (who isn't these days?), we ask that you time your payment accordingly.

1.6.1 Registering with CrypKey (Canada) Inc. to Get Your Developer Keys

Procedure

1. Finish testing CrypKey SDK and your product using the example developer keys.
2. Confirm you have paid for CrypKey SDK. If you selected BillNet30 or Telegraphic Transfer, please ensure your payment has been sent. We can authorize you to use CrypKey SDK on your product only after the payment has cleared.
3. Send us the following information via email, fax, or telephone:
 - a. your Customer Service Number (CSN), located on the inside cover of this user manual
 - b. the file name of your product (maximum 8.3 characters)
 - c. the password for your product (maximum 12 characters)
 - d. your site code from your Site Key Generator (see Section 1.6.3: Obtaining Authorization for Your Site Key Generator)

1.6.2 Using the Temporary License for Development

For the purposes of development and experimentation, you can temporarily use the following keys in your source code:

- **User Key:** D050 815C D1A2 A79D B103

- **Master Key:** 2A5D 57C4 1B4C 135B F09E 17F7 600B 2D70 79E8 F275 C36A

You do not require a temporary site key.

To use these keys, you must (temporarily) either place a file called example.exe in the same directory as your application or rename your executable to example.exe. Also, you must use “CRYPKEY” as your password whenever you use your Site Key Generator in test authorizations.

Under this temporary arrangement, you are checking and modifying the license for example.exe. This means you can use example.exe to test your program’s license operations.

1.6.3 Obtaining Authorization for Your Site Key Generator

In order to create site keys using your Site Key Generator, you must first provide us with the site code from your Site Key Generator (you can determine the site code by clicking **License** in the Site Key Generator dialog box: see below). We authorize your Site Key Generator once you finish testing CrypKey with your application.

Do not ship your application using the keys that are provided to you with the trial software. The example keys are strictly for use during your trial period, as anyone with the CrypKey product can create keys for any product protected with the example keys.

Procedure

When you have finished testing your product with CrypKey and are ready to acquire Site Key Generator authorization from us, use the following steps to determine the site code from your Site Key Generator:

1. Start your Site Key Generator by double-clicking its icon in the CrypKey SDK program group.

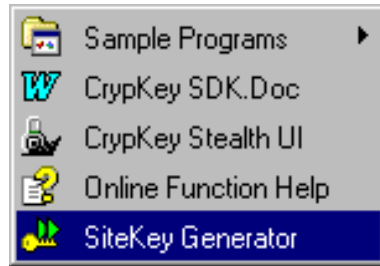


Figure 1: Site Key Generator Startup

2. Click **License** in the Site Key Generator dialog box. Your site code appears in the License dialog box, as shown below.

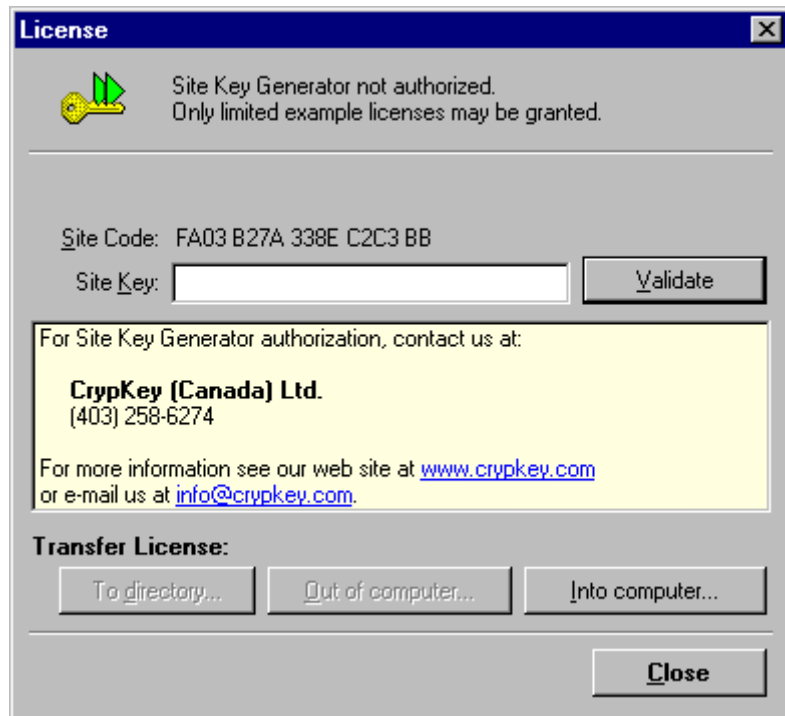


Figure 2: License Window – Not Authorized

3. Send us your site code via email, fax, or phone, along with the other information outlined in Section 1.6.1: Registering with CrypKey (Canada) Inc. to Get Your Developer Keys.
4. Once you receive your site key, type it (or cut and paste it from our email message) into the Site Key field in the License dialog box.

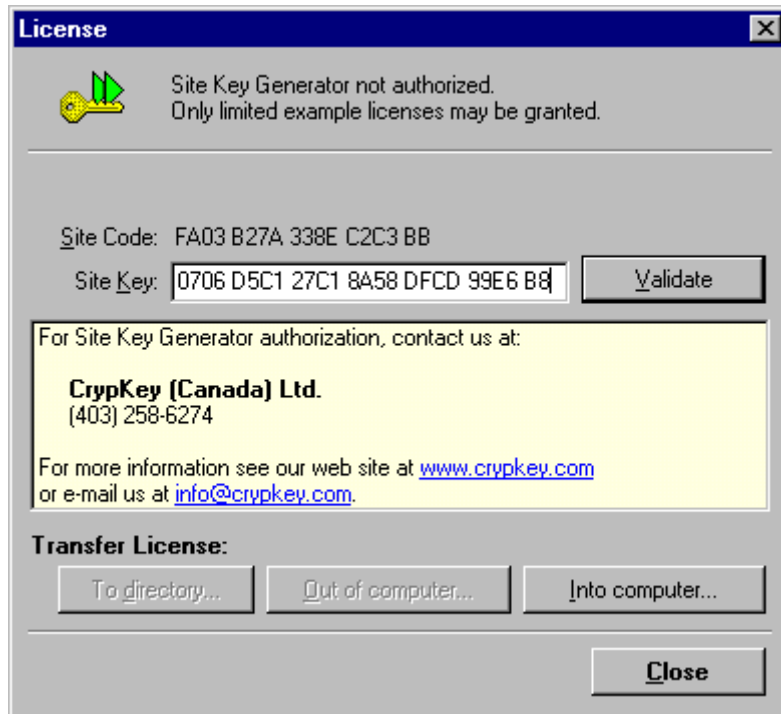


Figure 3: License Window – Site Key Inserted

5. Click **Validate**. To ensure that your Site Key Generator is authorized, return to the License dialog box:

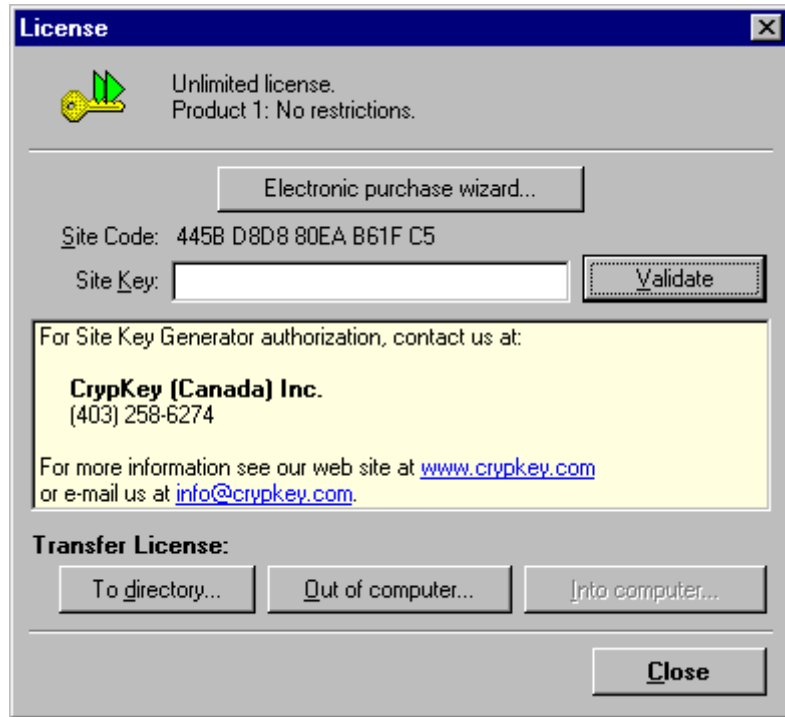


Figure 4: License Window – Authorized

1.7 CrypKey Licensing Implementation

The following steps describe how CrypKey is used in the development process:

1. **You (the developer) purchase the CrypKey software licensing system and program your application with the library security functions.** CrypKey can be best thought of as a Software Licensing API (SLAPI). You must embed the primary CrypKey functions in the source code using CrypKey SDK, which handles the initialization, license verification, site code generation, and termination features.

You may also choose to organize the features of your application in terms of specific “levels” and “options”, which are automatically facilitated within the CrypKey system. In fact, we recommend you do this if it is appropriate for the application, since this allows you to use the full feature set that CrypKey provides for licensing.

CrypKey Canada Inc. also provides a product called CrypKey Instant, that embeds the CrypKey licensing protection system within your application executable without requiring any changes to the source code. This product offers an alternative method of copyright protection to developers and distributors.

2. You experiment with CrypKey and test the application using the temporary license provided. If you are completely satisfied with CrypKey, fax us the required information. We fax you keys that activate your Site Key Generator license for the specific product(s) in development. The product executable name, password, Site Key Generator site code, and customer service identification number are required in order to create the developer keys that enable you to license the product(s) to customers using the Site Key Generator. There is a 30-day money-back guarantee, so you may fully test the CrypKey system, without risk, using the temporary license provided. All CrypKey SDK features are available to you during the trial period.

Please note that CrypKey cannot be returned once the developer keys for your product(s) have been issued.

3. You distribute the application that you have developed to potential clients. If the product functions initially in demonstration mode, the client can experiment with the product to determine what functions and options are required. You can provide prospective customers with a free copy of the application that has a 30-day limited license (this is configurable). This temporary license is fully protected by CrypKey and the application ceases to operate when this temporary license expires. Until then, the application can function normally and the client can test it out on his or her system. Note that program behavior, when it is not authorized, is completely under your control, however, a demonstration mode is recommended.
4. **When the customer is satisfied with the application, he or she obtains a site code from the application and transfers the code to you, typically by email, along with the license permissions the customer wants to purchase. The customer also provides payment to you.** The site code is based on a special internal hardware signature that is unique to the personal computer that the application is installed on. The site code appears as an encrypted alphanumeric number. CrypKey allows license restrictions to be specified in terms of a limited (or unlimited) number of days or runs. “Runs” are implemented in the application in whatever manner you designate. The number of program copies or concurrent network users is also specified within the license.

5. You enter the customer's site code into the Site Key Generator and set the license configuration. A site key, which provides specified access to the application, is then generated and given to the customer when you receive payment. The specific levels, options, and permissions are generally referred to as the "license" in the CrypKey system. This information is embedded and encrypted within the site key and it activates and configures the customer's license when it is entered into the application by the customer and decrypted. Any site key you generate works only with the installed application that created the site code provided by the customer. The Site Key Generator automatically verifies that the site code given is authentic.
6. Your customer uses the site key provided by you to unlock the application under the terms specified. Site codes and site keys can be transferred by email, telephone, or other means because they are alphanumeric strings of manageable length. The process of issuing a site key for a customer's specific site code is referred to as "authorization" in the CrypKey system. CrypKey automatically tracks restrictions in terms of days or runs and invalidates the license when the time period expires or the number of runs is used. The CrypKey functions inside the application verify that the site code/site key combination is valid before access is provided to the customer. Also, the license is uniquely specific to the personal computer on which the application runs. If the customer copies the program and its license files to another computer, the license becomes invalid on that machine and does not allow access. This occurs because CrypKey is capable of identifying the exact computer the license was issued for without the use of an external hardware (dongle) or floppy disk key.
7. **You achieve fame and fortune as customers everywhere are pleased with the product and the clean, non-intrusive licensing system you have provided.** Steps 4 through 6 can be repeated if the customer's needs change and the customer wants to buy licenses for different levels or options for the application.

1.8 New Features in Version 6.0

CrypKey, as a concept, has proven itself a hundred times over in the software marketplace. Our next step was to perfect CrypKey's performance. Our goal with the present release is to make CrypKey perform with a very high standard of security and robustness on all the platforms on which it runs. With version 6.0, we have succeeded in this goal.

CrypKey 6.0 has benefited from the experience of our now-sizable customer base. We have concentrated on tracking down and fixing every known problem on the many platforms CrypKey supports. The result is robust software. Before this release, 99% of all support problems were fixed just by sending this software.

We also have a related product, CrypKey Instant, that:

- protects DOS, DOS 32-bit, Win16, and Win32 programs *with no source code changes*
- does network floating licensing with no source code changes
- supports international languages and enables you to switch or add languages at runtime
- encrypts data

More information about CrypKey Instant is available on our website, at <http://crypkey.com>.

The changes we've made in CrypKey 6.0 include:

- CloneBuster™ technology for hard drive serial number (HDSN) recognition
- EasyLicense
- dynamic, multi-key encryption
- enhanced anti-hacking capability (StealthPLUS™)

These new features are described in more detail below.

1.8.1 CloneBuster™ Technology

Software developers are discovering they are losing billions of dollars in a worldwide massive license theft of large scale. Fueled by the recent surge in use of legal, low-cost hard drive (HD) cloning, or “ghosting”, software, easily-obtained \$49 utilities with names like Ghost and XXCopy are rendering once-popular copy protection techniques useless, generating fast lucrative returns for software pirates.

Many popular copy protection technologies that depend on placing some type of simple software footprint are now easily defeated by these new HD cloning techniques and are no longer an effective option. Many unproven, startup, venture-capital backed, rights management platform providers are not practically viable. They offer grand, and often

expensive, protection schemes that fail to recognize all the numerous firewall, integration, distributed deployment, and support issues generated by their so-called connected product vision. Hardware dongles provide limited protection against HD cloning, but are expensive and disliked by end users. CrypKey 6.0, which includes CloneBuster™ technology, effectively solves this problem.

Most modern IDE hard drives have burned-in serial numbers (HDSN). Not to be confused with the easily changed volume serial number, the HDSN is a permanent, read-only attribute of a personal computer hard drive. The HDSN is very difficult to access, but CrypKey SDK 6.0 with CloneBuster™ technology has this ability (patent pending). Leveraging CrypKey's ten-year, proven protection architecture, CloneBuster™ detects and logs the HDSN, model number, and firmware identifier and uses this information in its proprietary licensing algorithms. Although HD cloning programs succeed in breaking other simple copy protection schemes in order to copy an entire hard drive, they cannot achieve this against CrypKey armed with CloneBuster™.

1.8.2 EasyLicense

Some customers need the simplest possible form of copy protection when a particular application runs only on a particular computer. These customers do not want to deal with any associated issues of transfer, network license, trial period, or number of copies. Whether the software is offloaded and put back, deleted and restored later, or the entire hard drive is erased and restored, the application stills run on the computer.

EasyLicense offers this type of protection coupled with simplicity.

1.8.3 Dynamic Encryption

We have increased the bit size of our Site Code and Site Key encryption. Never satisfied with good enough, we have also created a dynamic multiple encryption key scheme, so that encryption keys are constantly changing on the fly. This is truly a code breaker's nightmare. This also halts any attempts to create a rogue key generator.

1.8.4 Enhanced Anti-Hacking

StealthPLUS™, our anti-hacking layer included with CrypKey, is a critical component as it stops the hacker from reverse engineering and circumventing the copy protection. StealthPLUS™ already encrypts your binary file and guards against program patches while it is in memory. We have enhanced StealthPLUS™ so it partially un-encrypts programs while it runs in memory and re-encrypt the parts that are not running. The

program teams up in memory with a special symbiotic process that un-encrypts parts as needed, dramatically decreasing vulnerability.

1.9 Updating to CrypKey SDK v6.0

To update CrypKey 6.0 from a previous version:

1. Obtain a new Master Key for each 32-bit application you wish to protect.
2. Obtain a new Site Key for your version 6.0 Site Key Generator.
3. If you have a 32-bit application, add a new file, CRP32001.NGN, to the directory your application resides in . This file is found in the \Libs\32bit directory that is created when you install CrypKey 6.0.

The version 6.0 libraries work only with the new 32-bit Site Key Generator 6.0 and later. Your old Site Key Generator cannot authorize these.

4. Please note the new Thunk library names CRYP95F.DLL and CRYP9516F.DLL. These new libraries must be present when Win9X and ME are run.

The name CRP9516F.DLL replaces all past versions of CRP9516?.DLL. The name CRYP95F.DLL replaces all past versions of CRYP95?.DLL. However, the name of the other file in the same directory, CK16RMV.EXE, has not changed.

5. The location of the Thunk DLLs has changed. They must be found in one of the following:
 - the directory (searched first) where your application resides
 - the directory (searched second) where your CrypKey license resides

If they do not reside in one of these directories, a THUNK_LIB_NOT_FOUND error results. Windows and system directories will not work.

Note: For VB and Delphi users—when you run your application under the IDE (not compiled standalone), the application directory is the IDE directory, not the directory where your application is. We previously told you to correct this problem by putting the Thunk libraries in the Win Directory, but this will not work now. Locate the Thunk libraries in the directory where you tell CrypKey to put its license.

When developing with the .NET or C++ IDEs you cannot debug calls to CrypKey. If you need to debug your programs, bypass your CrypKey calls temporarily.

6. This update includes all 6.0 libraries and DLLs. You need only those that apply to your application.
7. Do one of the following:
 - reconfigure SKW
 - copy your skw.ini from a previously configured SKW and add the following line at the end of your file:

```
EasyLicense=0
```

8. An uninstall feature, and other important changes, have been added to the NT Driver (see Sec. 5.6.4).

Installation

This chapter provides details on how to install CrypKey SDK and its associated network drivers on your system.

A

network driver does not have to be installed to use a CrypKey-protected application in standalone mode or to use CrypKey SDK on your machine (except in the case of Windows NT platforms—see Sec. 8: Network Licensing; and Sec. 5.6: Creating a Windows NT License Service Installation). Drivers are provided so you can run a CrypKey-protected application from a server on client workstations over a network.

2.1 Getting Started

You can install CrypKey SDK from a CD provided by CrypKey (Canada) Inc. or, if you are a CrypKey subscriber, by downloading the installation files from the CrypKey website. As you will see in Sections 2.1.1 and 2.1.2, the sequences of windows for the two types of installation are quite similar.

Whether run from a CD or from downloaded files, the installation program creates the following CrypKey group in your Start menu (Windows 95/98/ME/NT/XP/2000) or in Program Manager (Windows 3.1):

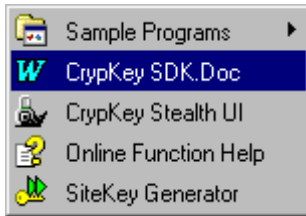


Figure 5: CrypKey SDK Program Group

2.1.1 Installing CrypKey SDK from CD

Procedure

1. Ensure Windows is running on your computer.

2. Insert the CrypKey CD-ROM into your CD-ROM drive. CrypKey automatically starts up, displaying the following window:



Figure 6: Opening CrypKey CD Installation Window

3. Click the **Install** button. The system displays the following window:



Figure 7: CD Installation Screen 1

4. Choose **Install CrypKey SDK**. The system displays the InstallShield Wizard, shown below. In this window, and the others in the sequence, simply click **Next** or **Yes**, whichever is presented, after you have responded to the prompts, if any, shown in the windows.

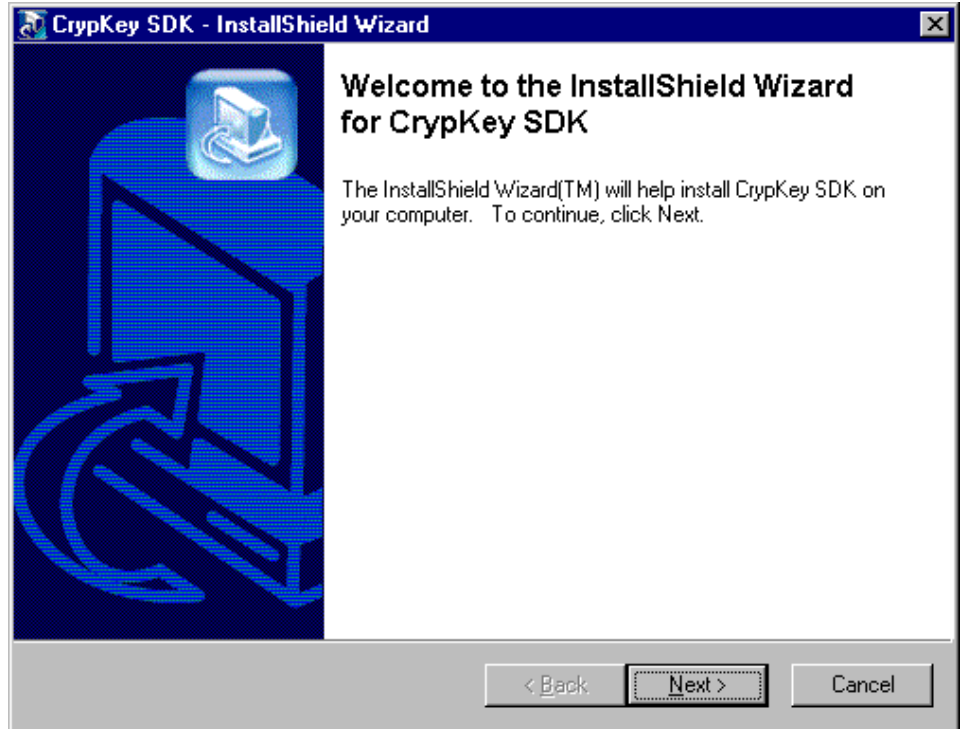


Figure 8: CD Installation Screen 2

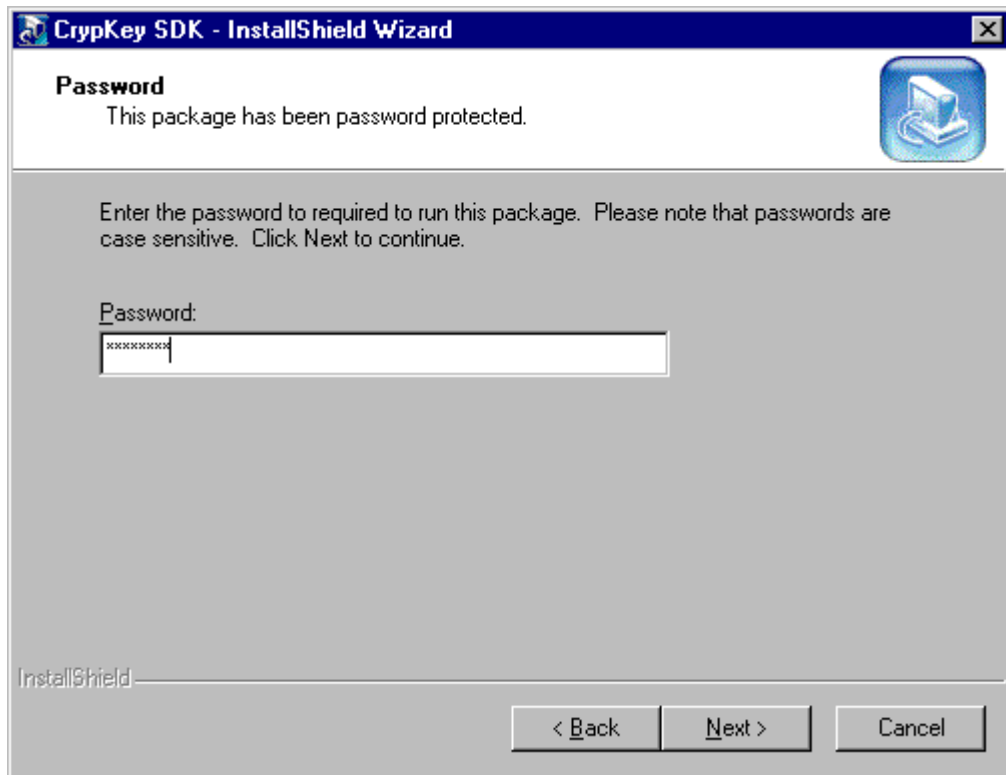


Figure 9: CD Installation Screen 3

4. In the above window, type the password provided by CrypKey. When you have correctly entered the entire password, the **Next** button is enabled, as shown above. Click **Next** to display the following window:

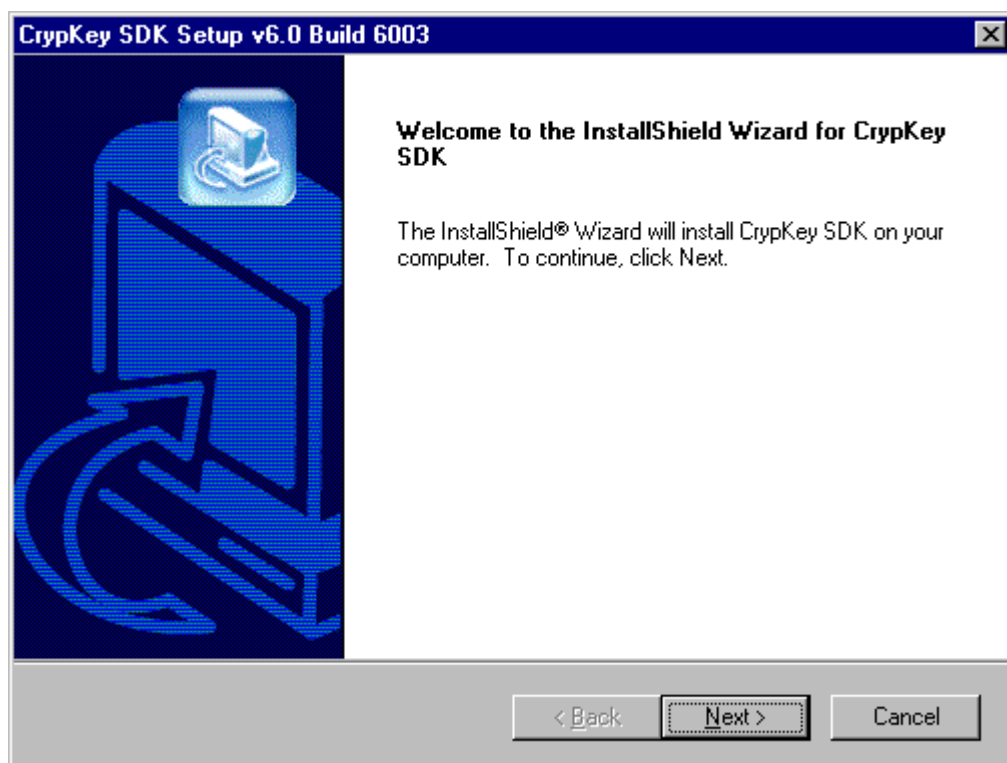


Figure 10: CD Installation Screen 4

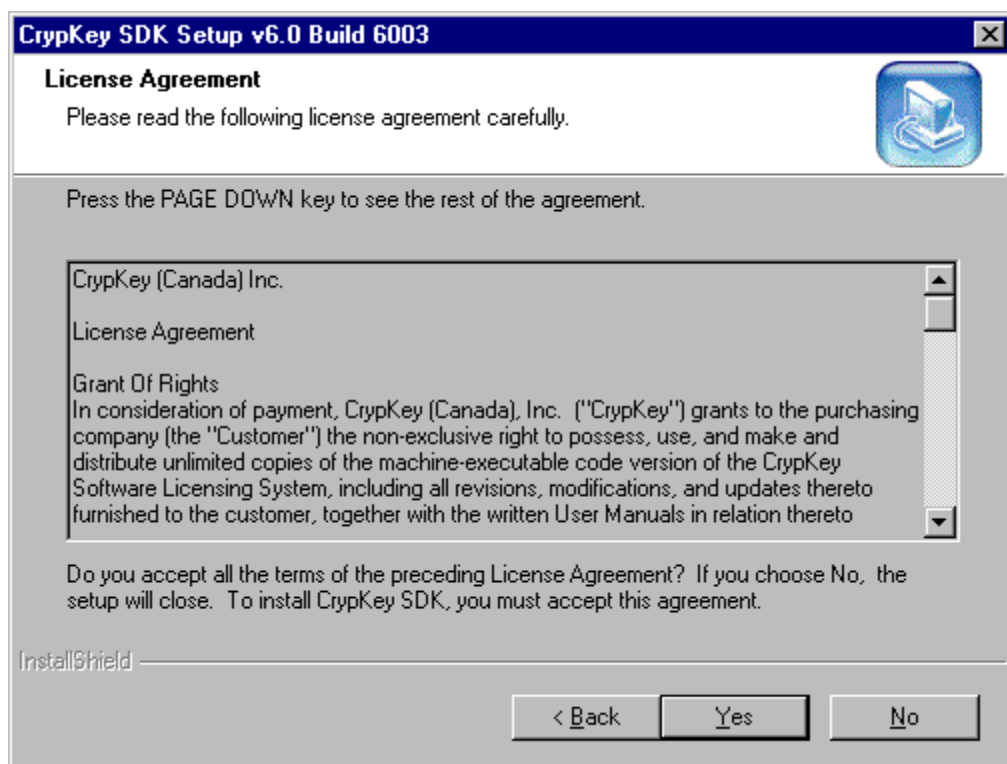


Figure 11: CD Installation Screen 5

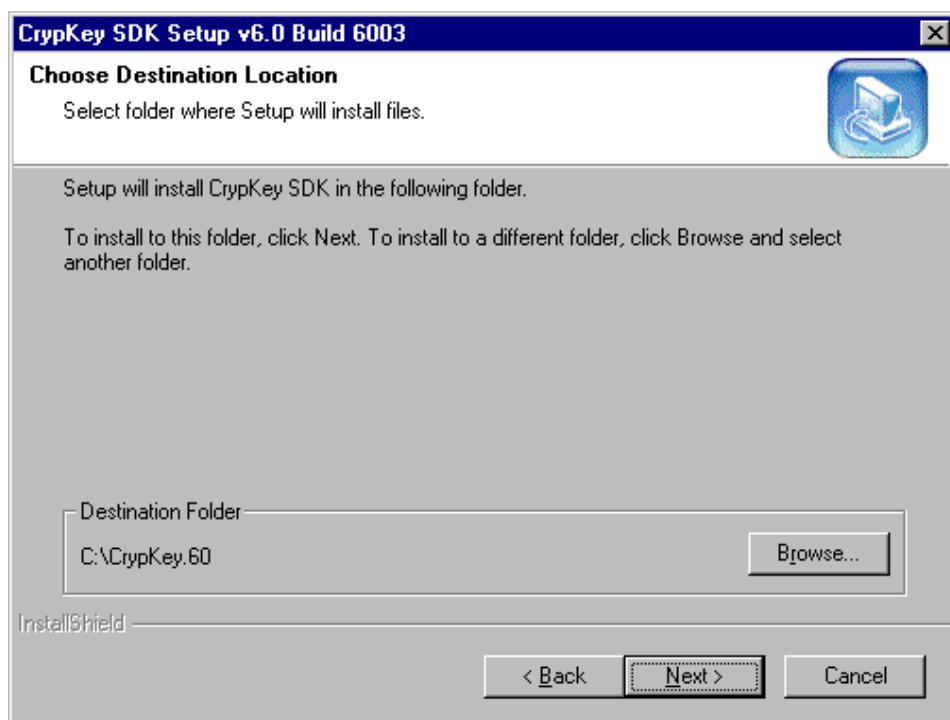


Figure 12: CD Installation Screen 6

5. In the above window, specify where you want CrypKey SDK installed by accepting the default directory provided by the installation program or by typing in a path for it to use. If the directory does not already exist, the installation program creates it for you.

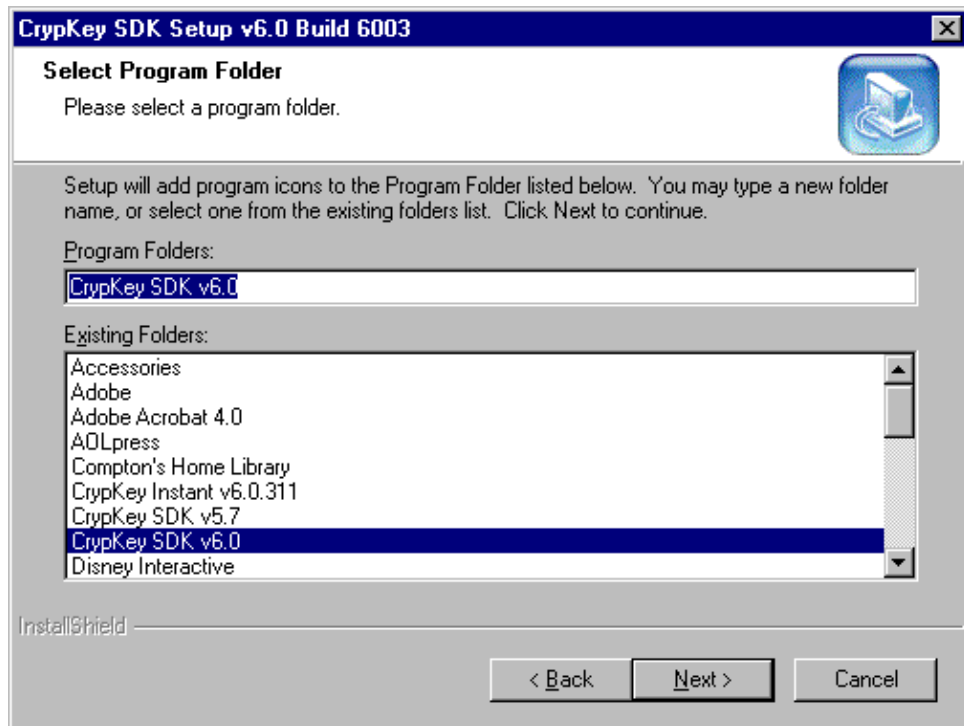


Figure 13: CD Installation Screen 7

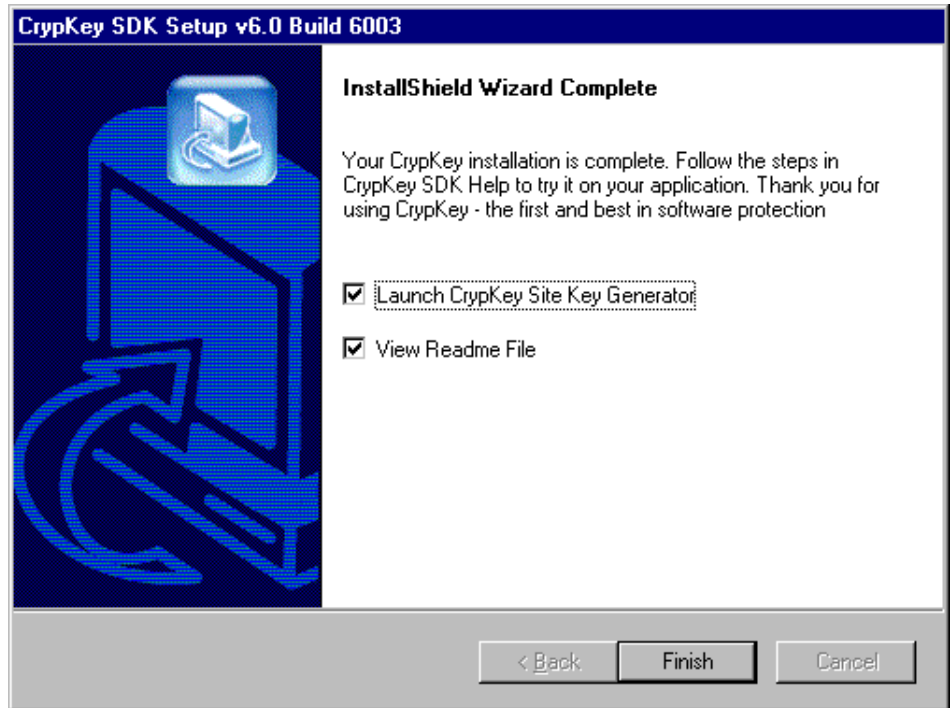


Figure 14: CD Installation Screen 8

6. In the above window, the options **Launch CrypKey Site Key Generator** and **View Readme File** are pre-selected. Click either checkbox to deselect it.

We strongly recommend that you read the **Readme** file immediately, as it may contain details that were not available in time to be included in this manual.

2.1.2 Downloading and Installing CrypKey SDK

Procedure

1. When you have completed the download of the installation files to your computer, double-click the file **CrypKeySDK60**.

The system displays the following window. This is the first of a sequence of standard installation windows. In each one, simply click **Next** after you have responded to the prompts, if any, in that window.

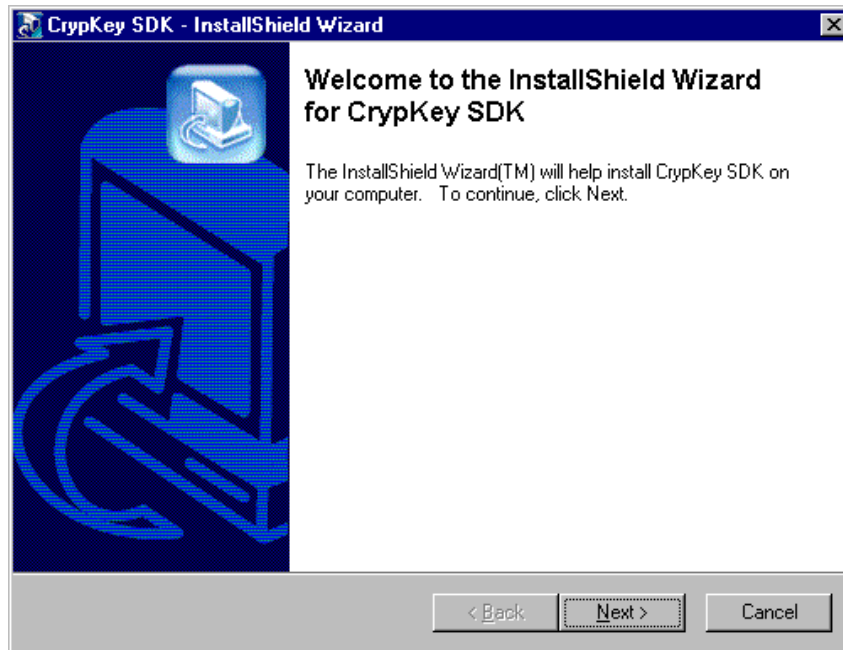


Figure 15: Download Installation Screen 1

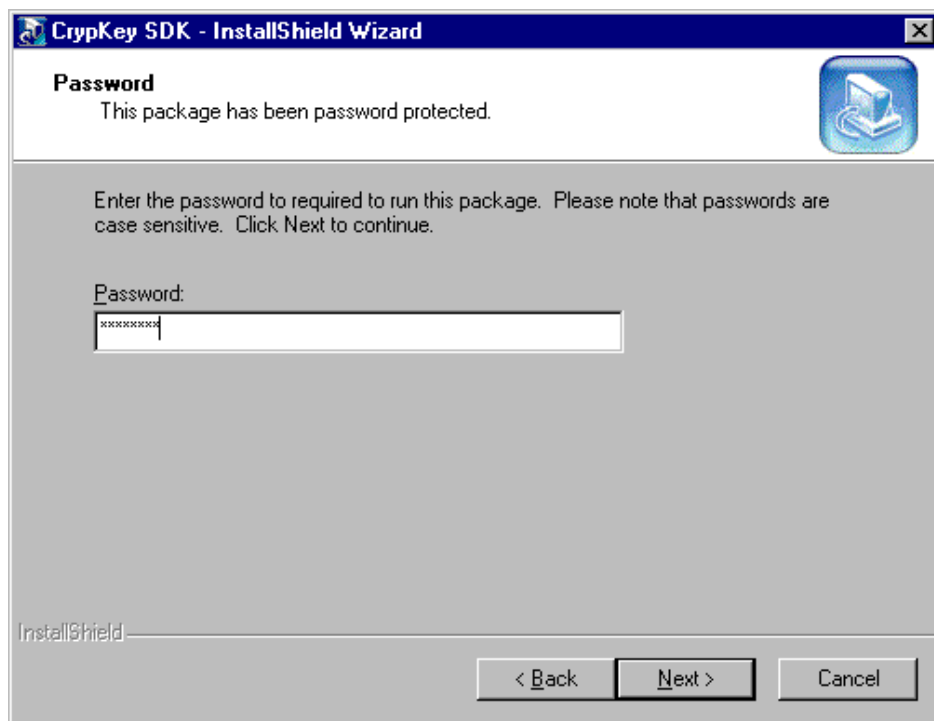


Figure 16: Download Installation Screen 2

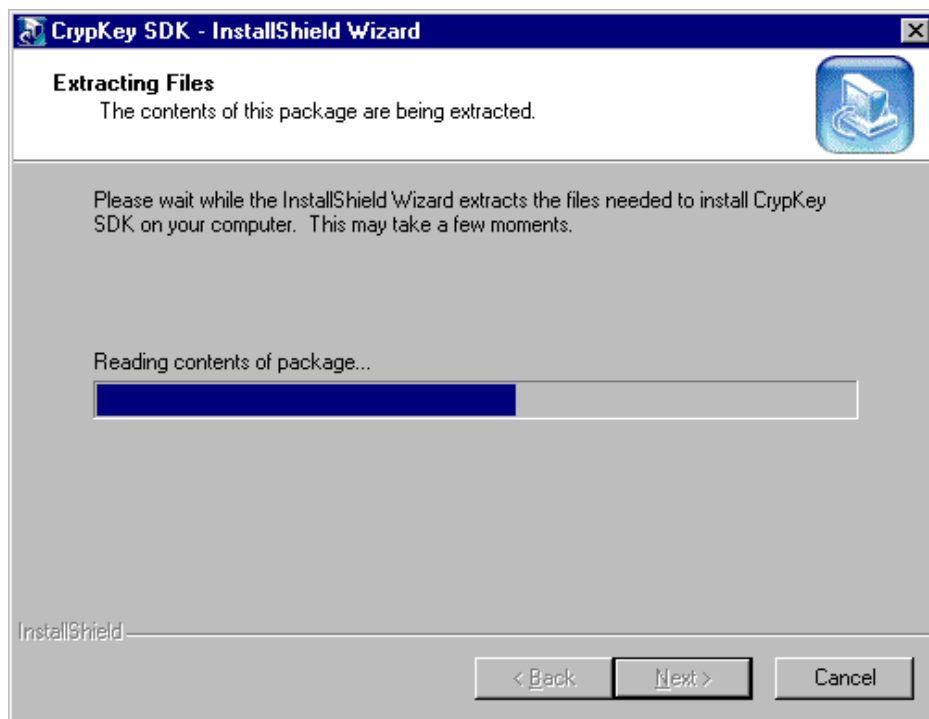


Figure 17: Download Installation Screen 3

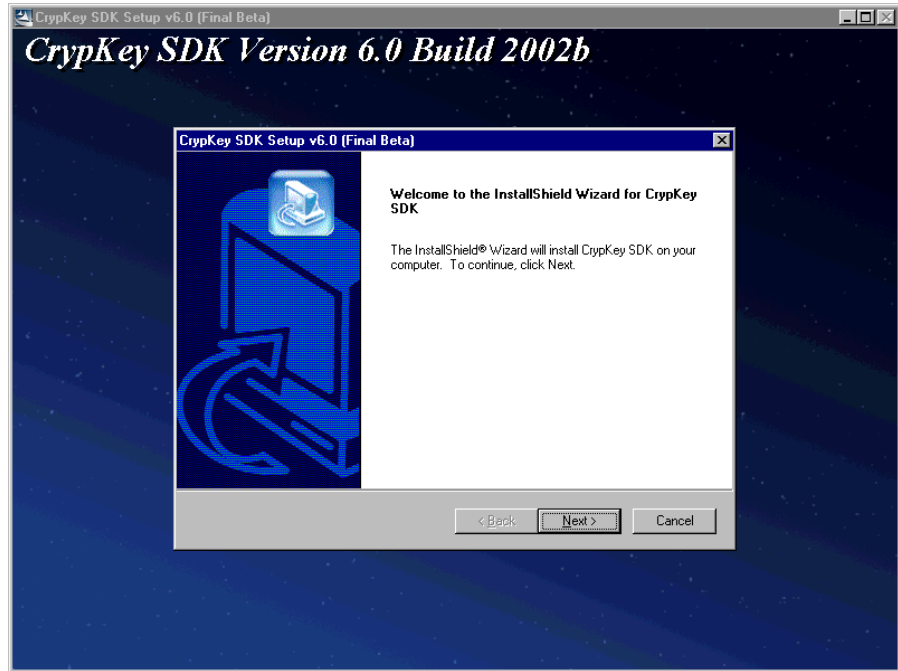


Figure 18: Download Installation Screen 4



Figure 19: Download Installation Screen 5

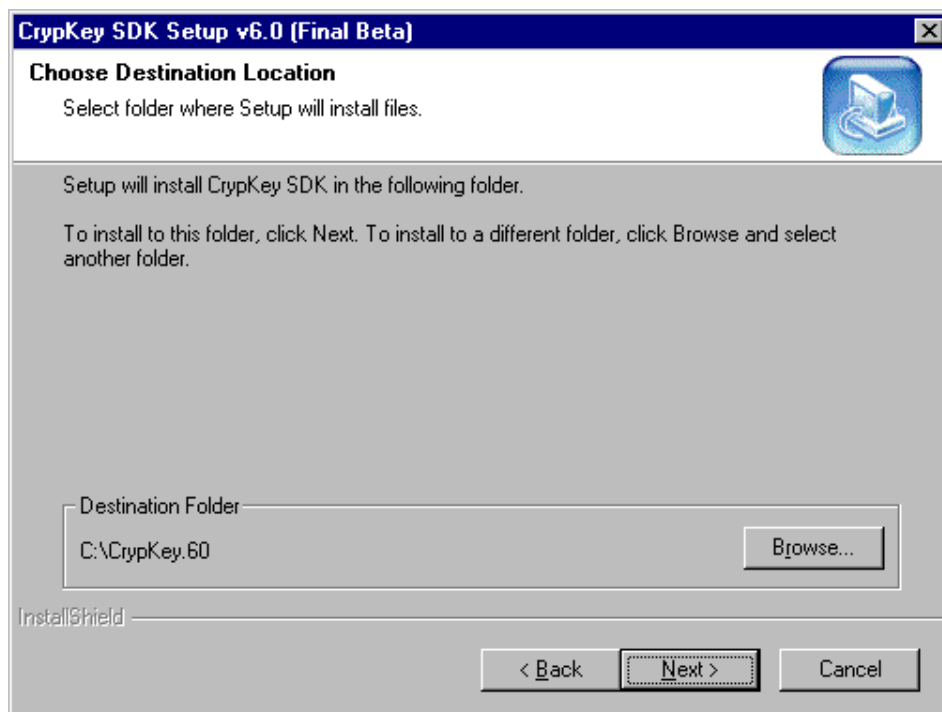


Figure 20: Download Installation Screen 6

2. In the above window, specify where you want CrypKey SDK installed by accepting the default directory provided by the installation program or by typing in a path for it to use. If the directory does not already exist, the installation program creates it for you.
3. Once the drive and directory are correct, click **OK** to continue the installation process.

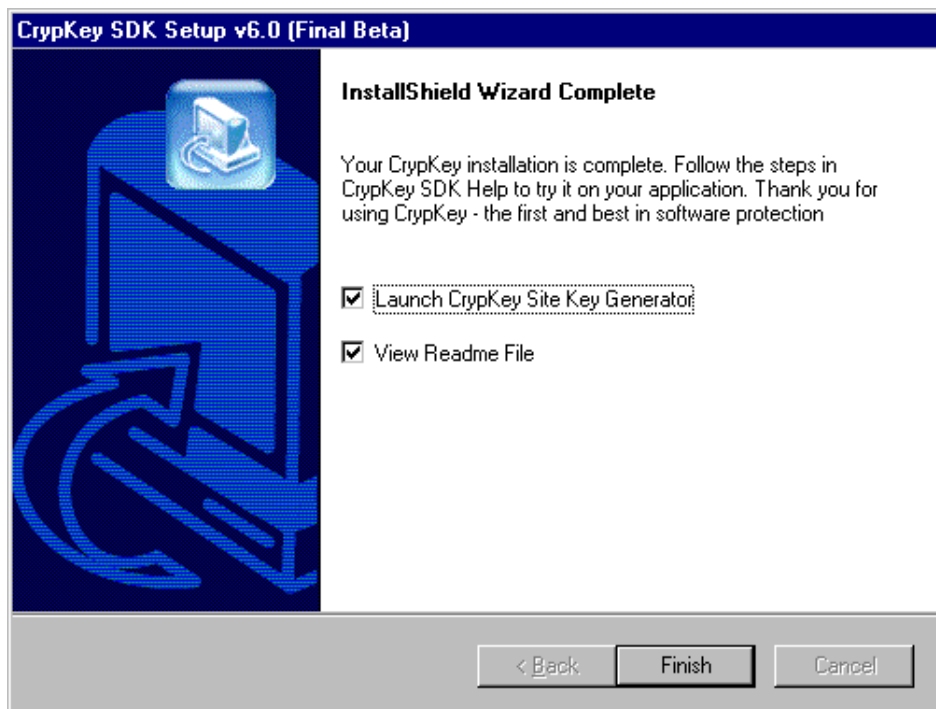


Figure 21: Download Installation Screen 7

4. In the above window, you are asked if you want to view the **Readme** file. You may read the file now or choose to view it later.

We strongly recommend that you read the **Readme** immediately, as it may contain last-minute details that were not available in time to be included in this manual.

2.1.3 Packing List and Readme Files

The packing list is a file (named packlist.txt) in your CrypKey directory. It lists all files in the CrypKey system and includes a short description of each one. You can use the packing list as a quick reference for information on the files included in your installation. This is especially useful if you need to find a library you want to use.

The **readme** file is located in your CrypKey directory. It contains important last-minute information that was not available in time to be included in the manual. You should read it after installing CrypKey on your system.

Following is the information in the **readme** file, at the time of publication, about the contents of each installed directory.

1. Demos

This directory contains the compiled executable versions of the sample code found in the Samples directory. If you are using NT, this directory will be set up to have the CrypKey NT Driver serving it.

2. External

This directory contains sample code for using External Linking with CrypKey. You would use this if you could not link to one of our libraries of DLLs from your particular development environment. (Example: Turbo Pascal).

3. Include

This directory contains the include files you would use if using C or C++. Currently, there is only `crypkey.h`.

4. Lib

This contains a number of different versions of the CrypKey library. To help you find the files you need, this directory is further subdivided into 16 and 32 bit. Under both 16 and 32 bit, there are libraries for Microsoft and Borland compilers, as well as generic 16-bit DLLs.

5. Netdrive

This directory contains the driver needed for all CrypKey NT Platform usage, as well as drivers for using CrypKey with networks.

Five server types are supported:

- NT Platform (Includes XP, Win200, NT)
- Windows 3.x or 9.x
- Novell 3.x and higher

- MSDOS
- OS2

6. Samples

This directory contains various sample code from different platforms. Some samples are contributed by kindhearted CrypKey customers (and we always welcome more). These samples are a good starting point and can always be improved upon.

7. Site Key Generator

This directory houses the CrypKey SiteKey Generator. For those updating from previous versions, we think you will like the improvements.

8. Stealth

This directory contains two versions of CrypKey StealthPLUS™:

- STELTHUI.EXE—user interface version
- STELTHCM.EXE—command line version

9. CrypKey SDK COM Object

This directory contains the CrypKey COM object. This is the easiest library to integrate into VB and Visual C. This directory also includes the documentation and VB and VC sample programs.

10. CrypKey SDK dotNET

This directory contains the CrypKey .NET assembly. This is the easiest library to integrate into the .NET languages (VB.NET, C#.NET, and C++.NET). This directory also includes the documentation and C#.NET and VB.NET sample programs.

11. This directory contains this manual. You may wish to use some parts in your software documentation.

2.2 CrypKey License Service Installation

This section pertains to all programs running on the NT File System (NTFS), including MS-DOS and Win16, as well as native Win32 programs. The CrypKey License Service must be installed on Windows NT/2000/XP operating systems.

The CrypKey License Service manages licenses. In NT environments, it is required for either standalone or network installations.

To run a CrypKey-protected MS-DOS, Win16, or Win32 program from a local drive on a Windows NT platform, you must install the CrypKey License Service on that machine. The CrypKey License Service is required because Windows NT does not allow programs to directly access the computer hardware, which CrypKey must do. For the installation procedures, see Sec. 5.6: Creating a Windows NT License Service Installation.

If you are currently running CrypKey on a Windows NT personal computer, you already have the CrypKey License Service installed. If you want to use the License Service with the installation of your application, you can find it in C:\CrypKey.60\Stealth.

StealthPLUS™ can install the License Service automatically for you. For more details, see Chapter 6: Protecting Your Software with StealthPLUS™.

We no longer support custom installations for Windows NT drivers.

2.3 Windows 9x/ME Network Installation

The file wckserve.exe is the Windows 3.1/9x CrypKey network communications driver. It is a Windows program that can be run in any Windows environment to allow the computer to act as a server for CrypKey-protected programs. Wckserve.exe runs in the background and is transparent to the operation of the computer.

Before running wckserve.exe, an environment variable must be set to tell it where the CrypKey-protected program is. The best way to do this is to put a command in the autoexec.bat file as follows:

```
SET CKSERVE=dir; [dir2;] [dir3;]
```

Here, dir, dir2, dir3, etc., are paths to the directories that hold CrypKey-protected programs. If there is more than one CrypKey-protected program on the computer, a path must be given for each directory, ending with a semicolon. When the computer is rebooted, the SET command takes effect with the parameters specified.

Alternatively, wckserve.exe accepts the directory string normally in the WCKSERVE environment as a command line variable:

```
WCKSERVE.EXE C:\MYAPP1;C:\MYAPP2
```

Here, MYAPP1 and MYAPP2 are directories containing CrypKey-protected applications.

To configure wkserve.exe to automatically start and run in the background, add wkserve to the Startup group. When Windows is restarted, the driver runs.

Note: the server must give read and write permission to its clients for the directory where the CrypKey-protected software resides. If write permission is not given, all network access requests are denied.

2.4 MS-DOS Network Driver Installation

Ckserve.exe is a terminate and stay resident (TSR) driver program that must be run on any networked MS-DOS computer that makes a CrypKey-protected program available to other computers. This driver is intended for MS-DOS servers and cannot be run on a Windows server. Ckserve.exe runs in the background and is transparent to the operation of the computer.

Before running ckserve.exe, an environment variable must be set to tell it where the CrypKey-protected program is. The best way to do this is to define an environment variable in the autoexec.bat file as follows:

```
SET CKSERVE=C\:\MYAPP1; C\:\MYAPP2; ...
```

Here, MYAPP1, MYAPP2, etc., are directories containing CrypKey-protected applications. If there is more than one CrypKey-protected program on the computer, a path must be given for each directory, ending with a semicolon. When the computer is rebooted, the SET command takes effect.

Alternatively, ckserve.exe accepts the directory string normally in the CKSERVE environment as a command line variable:

```
CKSERVE.EXE C:\MYAPP1;C:\MYAPP2; ...
```

Here, MYAPP1, MYAPP1, etc., are directories containing CrypKey-protected applications.

To start ckserve.exe, ensure the program is loaded on the computer. Type the program name from the MS-DOS command line with no options. This command should be put into the autoexec.bat file. You can put ckserve.exe in any directory, as long as you name the full path when you start it.

This program must be run from MS-DOS (version 3.1 or higher). It cannot be run with Windows on the same computer. Contact us if you need this ability.

The server must give read and write permission to its clients for any directories where CrypKey-protected software resides. If write permission is not given, all network access requests are denied.

2.4.1 Example

Consider an example where there are CrypKey-protected programs in the following directories:

C:\APPS\BINGO

E:\WINDOWS\WHIZBANG

D:\TEMP

If you want your users to be able to operate the first two programs from other computers that are linked by a network and you have ckserve.exe loaded into the directory C:\util, put the following commands in the autoexec.bat file:

```
SET CKSERVE=C:\APPS\BINGO;E:\WINDOWS\WHIZBANG;D:\TEMP
```

```
C:\UTIL\CKSERVE.EXE
```

Reboot the computer to put these commands into effect.

2.5 Novell NetWare Installation

CrypKey includes one Novell NetWare Loadable Module (NLM), which must be loaded into a NetWare server machine that is hosting CrypKey-protected programs with floating licenses.

Ckserver.nlm is used for all NetWare servers. It must be given read, write, filescan, modify, and delete permissions for the server machine that it is servicing. Ckserver.nlm can service only the local file server that it is installed on; this includes the SYS volume and all other volumes on the server.

Ckserver.nlm is self-configuring and detects any CrypKey-protected program on the server it runs on. It does not require any interaction or configuration from the administrator once it has been loaded.

2.6 Libraries and Distribution Files

Following are lists of 16- and 32-bit libraries from which you must distribute the appropriate files with the product that you provide to your customer. For convenience, both .LIB and .DLL files are listed here, although .LIB files are not distributed with your application. The.DLL files that must be distributed are indicated below; others are for distribution as required by your application. These libraries reside in your CrypKey.60\Lib directory.

For more details on linking your application to libraries, see Sec. 5.3.2: Linking.

For more details on libraries to be distributed for various platforms, see Sec. 14.3: CrypKey 6.X OS File Distribution Matrix.

See Sec. 2.6.3 for a discussion of the Thunk libraries within the 32-bit libraries.

2.6.1 16-Bit Libraries

The following libraries are found in ~\CrypKey.60\Lib\16Bit.

LCRYPKY7.LIB—Microsoft C 7.0 and DOS

LCRYPKYM.LIB—Microsoft C 7.0 and Windows 3.1

LCRYPKYD.DLL—Windows Dynamic Link Library (DLL)

LCRYPKYD.LIB—Microsoft C 7.0 Import Library for DLL

2.6.2 WIN32 Libraries

The following libraries are found in ~\CrypKey.60\Lib\32Bit.

Files required to be distributed

Table 1: Files required for distribution with 32-bit Applications

CRYP95f.DLL	These three files must be distributed
-------------	---------------------------------------

CRP9516f.DLL	with your program. They must be put in the same directory as your 32-bit application, or in the directory that holds your license.
CRP32001.NGN	

Files required for various purposes

CRP32M60.LIB—Microsoft VC++ 6.0 Win32 library (multiple thread Static RT Libraries)

CRP32D60.LIB—Microsoft VC++ 6.0 Win32 library (multiple thread Dynamic RT Libraries)

CRP32S60.LIB—Microsoft VC++ 6.0 Win32 library (single thread RT Libraries)

CRP32M42.LIB—Microsoft VC++ 4.2 Win32 library (multiple thread Static RT Libraries)

CRP32D42.LIB—Microsoft VC++ 4.2 Win32 library (multiple thread Dynamic RT Libraries)

CRP32S42.LIB—Microsoft VC++ 4.2 Win32 library (single thread RT Libraries)

CRP32DLL.DLL—Win32 DLL (multiple thread RT Libraries)

CRP32DLL.LIB—Microsoft VC++ 4.2 import lib for CRP32DLL.DLL

CRP32BMT.LIB—Borland C++ 4.51 Win32 library (multiple thread)

CRP32BST.LIB—Borland C++ 4.51 Win32 library (single thread)

The following files are found in other directories:

- in ~\CrypKey.60\CrypKey SDK COM Object -
CRYPKEYCOM.DLL—COM object for all compilers that can use COM objects.
- in ~\CrypKey.60\CrypKey SDK dotNET -

CRYPKEYNET.DLL—.NET assembly for all compilers that can use .NET assemblies.

2.6.3 32-bit Thunk Library Installation

A 32-bit CrypKey-protected application must include special CrypKey “thunk” libraries in its installation routine so that the application runs. These libraries are placed in the same directory location as the program. For 16-bit CrypKey-protected applications, these files are not needed.

CRYP9516f.dll, CRP95f.dll, and CK16RMV.exe are the three files that must be included for 32-bit applications in Windows 9x/ME (the first two files are thunk libraries. These libraries are not required if the program is installed on Windows NT, but to make your installation process consistent, you can include them anyway.

The location of the Thunk DLLs has changed. They must be found in one of the following:

- the directory (searched first) where your application resides
- the directory (searched second) where your CrypKey license resides

If they do not reside in one of these directories, a THUNK_LIB_NOT_FOUND error results. Windows and system directories will not work.

Note: For VB and Delphi users—when you run your application under the IDE (not compiled standalone), the application directory becomes the IDE directory, not the directory where your application is. If the thunk libraries are in your application directory, they will not be found by CrypKey. We previously told you to correct this problem by putting the Thunk libraries in the Win Directory, but this does not work now. You should put the Thunk libraries in the directory where you tell CrypKey to put its license.

Extra CRC security is added to the Thunk libraries, preventing file sizes from changing.

In previous CrypKey releases, when multiple versions of the program used the Thunk DLLs, or when CrypKey-protected applications ran, these DLLs were shared and sometimes crashed. In this version, the DLLs are **not** shared and consequently the DLLs don’t crash. Although there is still only one copy of the DLLs, CrypKey automatically creates and runs a copy of the DLLs for each program.

2.7 Connectivity to Microsoft Access

The rules enabling CrypKey to work with Microsoft Access are the same for both Windows 97 and Windows 2000:

1. The three DLLs Crp32dll.dll, Cryp95f.dll, Crp9516f.dll must be in the same directory.
2. This directory must be on the path or explicitly named.

You can achieve this in three ways:

- by putting the DLLs in a windows or system directory

Note: it is always better to keep your files out of common directories. You can avoid using the sys or win directories by choosing one of the other methods

- by adding your directory to the path and rebooting
- by always putting your files in the same directory, e.g. C:\Dan\Danstuff, and changing the declares in VB accordingly. For example from "Crp32dll.dll" to "C:\Dan\Danstuff\Cryp32ddll.dll"

Each method has merits and drawbacks, so it is up to you to choose the one to use.

Demonstration Programs

Demonstrations are provided to enable you to test and evaluate the CrypKey system in operation. There are two demonstration programs included with CrypKey SDK 6.0 — WEXAMPLE.EXE and EXAMPLE.EXE.

W

EXAMPLE.EXE is a Win32-based CrypKey-protected program that demonstrates all of the basic features of the CrypKey system. EXAMPLE.EXE is an MS-DOS based, CrypKey-protected program with no graphical user interface (GUI). The complete source code for both programs is included in the CrypKey SDK installation.

3.1 Authorizing a Windows Program

After installation, your Windows Program Manager contains a new program group with several new icons, as shown:

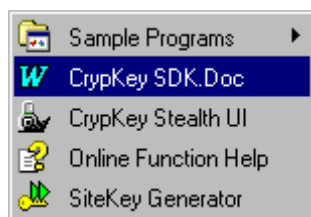


Figure 22: Program Group

One of the icons pointed to by the **Sample Programs** option, **Win32 Example**, launches the WEXAMPLE program, which demonstrates the authorization process. It is a useful tool that you can use to verify or troubleshoot CrypKey functions.

As in the normal authorization process for your products, the Site Key Generator is an essential element of the WEXAMPLE demonstration. For more details on how to use the Site Key Generator, see Sec. 4: Site Key Generator.

Procedure

1. Using the **Start** button, display the CrypKey program group shown in Figure 22.
2. Click the **Sample Programs** option. Your computer displays the following popup window:

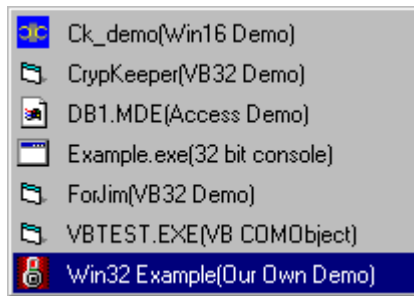


Figure 23: Sample Programs Group

3. In the above window, double-click the **Win32 Example** icon to start the WEXAMPLE.EXE program.

Your computer displays the TESTAPP window as shown:

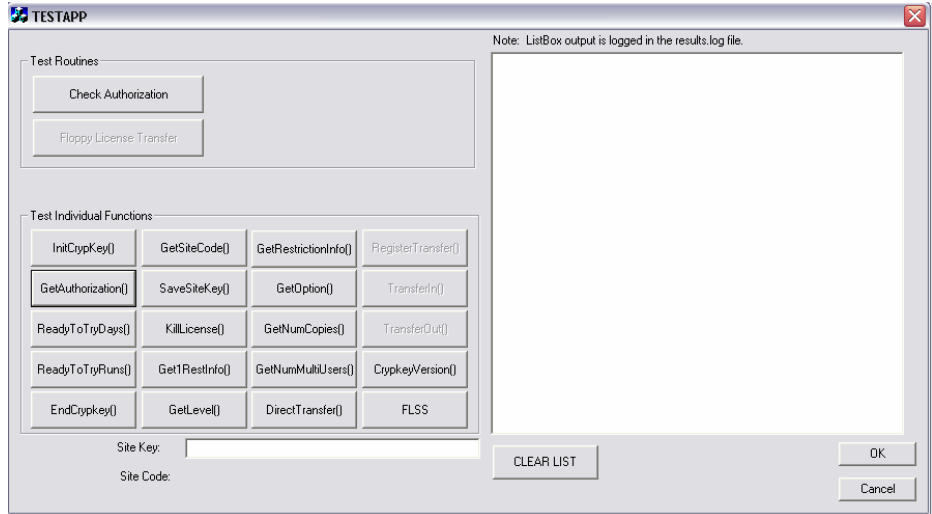


Figure 24: Opening WEXAMPLE.EXE (TESTAPP) Window

Note: the buttons in the TESTAPP window perform a number of CrypKey functions. You can use these buttons to troubleshoot the operational status of your CrypKey installation.

4. Click the following buttons in sequence: **InitCrypKey()**, **GetAuthorization()** and **GetSiteCode()**. As you click these buttons, the following messages appear in the window panel on the right side of the TESTAPP window:

```
InitCrypkey() called ...
INIT OK
```

```
GetAuthorization() called...
Oplevel = 0
AUTHORIZATION NOT PRESENT
```

```
GetSiteCode() called ...
SITE CODE OK
740A D598 71EC E8E3 35
```

The TESTAPP window appears as follows:

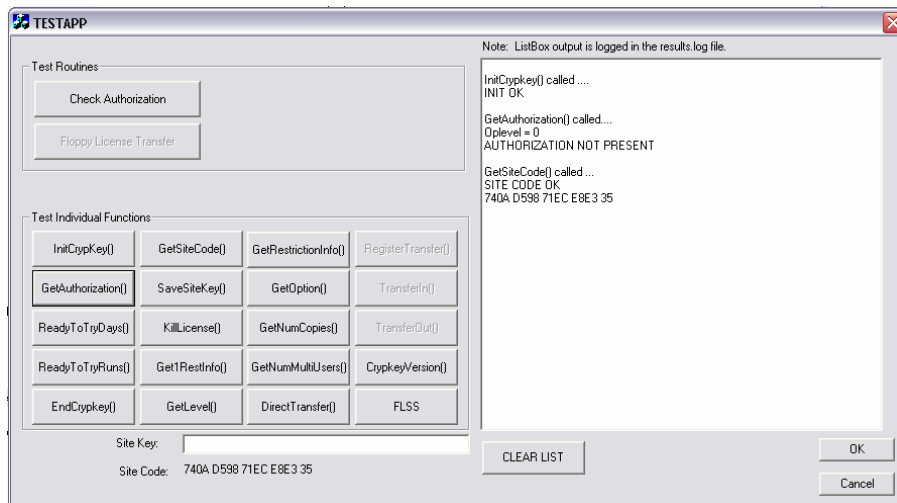


Figure 25: TESTAPP Window with Site Code

In the above window, note that the site code, shown in the returned messages panel, is also displayed below the **Site Key** field. For explanations of how the Site Code and Site Key are generated, and their purpose, see Sec. 4: Site Key Generator.

5. Select the Site Code located below the **Site Key** field and press [Ctrl]+[C] to copy the Site Code to the clipboard.
6. Double-click the **Site Key Generator** icon to start the `skw.exe` program.

You can use the [Alt]+[Tab] keys to toggle between the two program windows that are now active (TESTAPPS and the Site Key Generator).

Note: To place things in context... In the TESTAPPS window, you are playing the role of your customer by opening an application that you as the vendor have protected using CrypKey. In the Site Key Generator window, you are acting as yourself by receiving the site code generated by the CrypKey program bundled with WEXAMPLE.EXE and entering the code in order to generate a Site Key.

7. In the Site Key Generator window, press [Ctrl]+[V] to paste the site code (which you copied to the clipboard in Step 5) into the **Site code** field and then click the **Check** button to verify the site code. If the code is valid, the window shows the following:

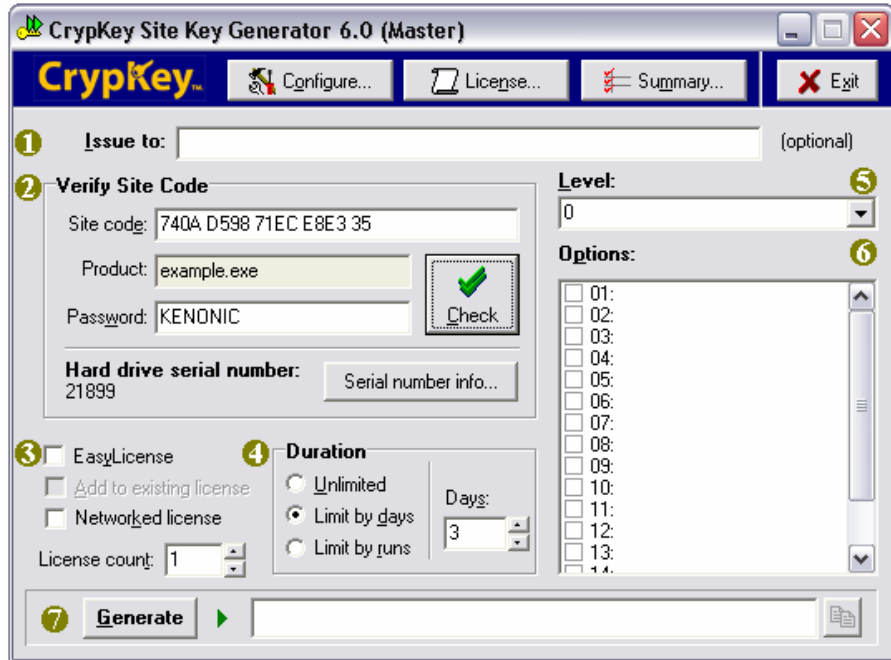


Figure 26: Site Key Generator Window with Validated Site Code

8. In the above window, enter the level, options, license type, and license restriction as desired. These items are explained in detail in Section 4: Site Key Generator.

9. Click the **Generate** button. The program generates a site key based on the site code and places it in the text box beside the **Generate** button.

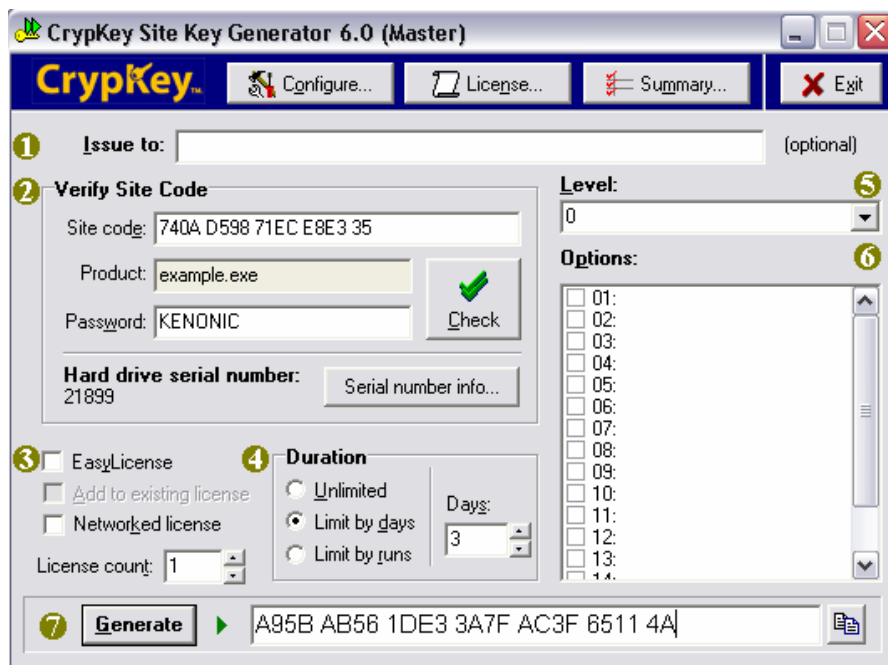


Figure 27: Generated Site Key

10. Select the generated site key in the above window and press [Ctrl]+[C] to copy the site key to the clipboard.
11. Toggling back to the TESTAPPS window, place the cursor in the **Site Key** field and press [Ctrl]+[V] to paste the generated site key into the **Site Key** field.

Note: Again, to place things in context... You are playing the role of your customer by receiving the site code from the vendor and entering the code into WEXAMPLE. Using the CrypKey function buttons in the TESTAPPS window, you now simulate product authorization.

12. Click the **SaveSiteKey()** button. The following return message is added to the preceding messages in the right panel of the TESTAPPS window:

```
SaveSiteKey called ...
SITE KEY ACCEPTED
```


13. Click the **GetAuthorization()** button. The following return message is added to the preceding messages in the right panel of the TESTAPPS window:

```
GetAuthorization() called ...
Oplevel = 0
AUTHORIZATION OK
```

The TESTAPPS window appears as follows:

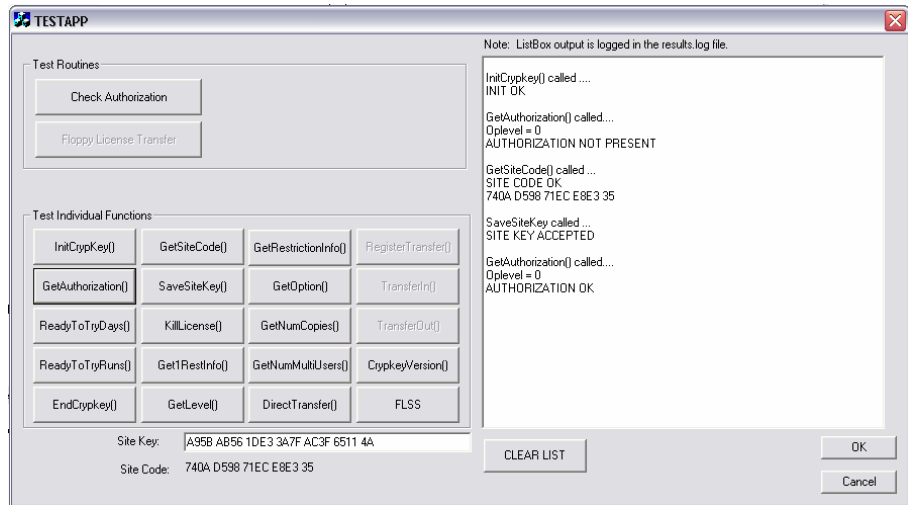


Figure 28: TESTAPPS Window with Authorized Site Key

You have just granted the demonstration program, WEXAMPLE.EXE, the privileges you assigned using the Site Key Generator. In an actual implementation, these privileges apply only to a specific copy of your application. You can send this authorization to a customer located anywhere in the world by email or telephone.

3.2 Authorizing an MS-DOS Program

After installation, your Windows Program Manager contains a new group with several new icons. One icon, Example, is an MS-DOS program that simply shows its current state of authorization. It represents your software, at your customer's place of business, for a demonstration. Another icon, Site Key Generator, is the actual program you use at your place of business to authorize your customer's programs.

Procedure

1. Click Windows **Start > Programs > CrypKey SDK v6.0**.

The system displays the following program group popup window:

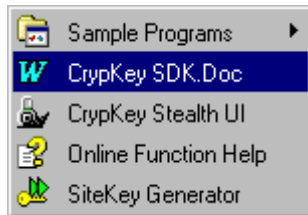


Figure 29: CrypKey SDK v6.0 Program Group Popup Window

2. In the above popup, double-click on the **Example** icon to start the example.exe program.
3. Press the “a” key to activate the Authorization option.
4. Write the site code on a piece of paper.
5. Double-click the Site Key Generator icon to start the skw.exe program. Use [Alt]+[Tab] to toggle between the two active program windows.
6. Type the site code into the Site Code field of the Site Key Generator.
7. Enter “CRYPKEY” into the password field.
8. Enter the level, options, license type, and license restriction as desired and click the **Validate** button. These items are explained in more detail in Section 4: Site Key Generator.
9. Write the site key on a piece of paper and press [Alt]+[Tab] to return to example.exe.
10. Type the site key into example.exe to authorize the program.

You have just granted example.exe the privileges you assigned. In an actual implementation, these privileges apply only to a specific copy of your application. You can fax or telephone this authorization to a customer anywhere in the world.

Site Key Generator

CrypKey SDK's Site Key Generator enables you to authorize clients to use your product.

Integral to CrypKey SDK, the Site Key Generator creates the keys that unlock your product. You can limit the number of days or runs over which clients can use your product or implement the unlimited usage option. You can also specify the number of program copies that can be made and the number of concurrent network users that are allowed. In addition, you can review previous transactions and sort them by date, customer or product name, and review archived log files.

4.1 Levels and Options

CrypKey SDK's levels and options allow you to authorize different features of your product. For example, if your product has multiple components and you want your clients to pay per component, toggle on only those components your clients want when you issue the site key.

The combined total number of bits that can be used for CrypKey levels and options is 32. However, you control what portions of this total are assigned to levels and options. For example, you could implement 28 options (each option uses one bit) for your product, keeping in mind that only four bits would remain for implementing levels.

Within this framework, you set a range of levels and options for each product using the Site Key Generator Configure tab page (see 4.3.2: Configuring Your Products). When your customer requests a license authorization for a product, use the Site Key Generator to:

- select a level (only one of the available levels) and options specifying what access you are providing to the customer for that product
- issue a site key to the customer, which incorporates the selected level and options

The customer enters the site key into your protected application on his or her computer. The levels and options information encoded into the site key become a permanent part of the customer's license, which defines the customer's privileges for the application.

4.1.1 Levels

With levels, you can choose one feature or style you want your clients to have. For example, you can divide your software into beginner, intermediate, and expert levels.

A level is an integer number that controls a variety of features in your product. For example, you may devise the following definitions:

- Level 0: no additional modules can be used
- Level 1: the graphic module can be used
- Level 2: the graphic and printing modules can be used



Levels can also be used for version control. For example, you may decide that the first version (v1) of your software runs regardless of level, while v1.1 requires Level 1 and v1.3 requires Level 2. Level strategies relating to versions are numerous, but you can choose one level at a time. Configure levels into the Site Key Generator using the

Configure button.

4.1.2 Options

With options, you can toggle on or off one or more different features of your product.

An option is a single bit used like an on/off switch. For example, if you want to offer clients a module that allows your product to perform printouts for an additional fee, you create an option similar to this:

- Option 1 (on): the module can be run
- Option 0 (off): the module cannot be run (a message appears noting that this option has not been purchased)



Up to 32 options can be used in CrypKey SDK. To create a site key enables options, use the Site Key Generator's **Configure** button.

4.2 Additional Features of the Site Key Generator

In addition to Levels and Options, the Site Key Generator offers flexible restrictions on licenses, EasyLicense for the user's convenience, and CloneBuster™ technology to prevent hard-drive cloning or “ghosting”.

4.2.1 License Duration Settings

You can configure a license using one of three different duration types, all of which can be transferred via direct transfer or floppy disk:

- **Runs:** allow clients to run your product for a specific number of runs (maximum: 32,768)
- **Days:** allows clients to run your product for a specific number of days (maximum: 32,768)
- **Unlimited:** allows clients to run your product without restrictions

When you are authorizing distributors (see Sec. 4.4: Distributor Authorizing License), the `RUNS_ONLY` and `DAYS_ONLY` options are available. When you enable `RUNS_ONLY` and select the runs license, the slave Site Key Generator can only create runs licenses. In addition, the total number of runs the generator grants a product is decremented from its own license count. Therefore, if a slave Site Key Generator is awarded a runs license with 1,000 runs and the `RUNS_ONLY` option is enabled, the generator has 800 runs left if it issues 200 runs to a product. The same theory holds true for the `DAYS_ONLY` option and days license.

Clock Manipulation Checks

There are programs available on the market that are designed to try to deceive time-restricted software protection systems. These programs automate the process of changing a computer's clock in order to bypass time-restricted protection. These programs change the time registered by the protected program. For example, if the registered time never extends beyond your program's trial period, a user can in theory run your program indefinitely.



CrypKey safeguards against clock manipulations by monitoring your customers' computers for the inconsistencies generated by such an attack. CrypKey detects when time is falsely reported to your program and prevents your program from running. Time reported by CrypKey is that of the server, adjusted to the local time zone of the computer.

CrypKey may also detect clock manipulation if users adjust their computer clock by more than several hours, even for benign purposes such as testing. If your program is prevented

from running in these situations, your customer can correct the situation by rebooting the computer.

4.2.2 EasyLicense

Some customers need the simplest possible form of copy protection when a particular application runs only on a particular computer. These customers do not want to deal with any associated issues of transfer, network license, trial period, or number of copies. Whether the software is offloaded and put back, deleted and restored later, or the entire hard drive is erased and restored, the application still runs on the computer. EasyLicense meets these requirements.

The features of EasyLicense include:

- Clients can back up their licenses and restore them later.
- The license files can be moved without affecting the license itself.
- The license transfer ability is disabled to prevent abuse.
- The only license type available for EasyLicense is an unlimited license.

EasyLicense works on any system that supports CrypKey's HDSN technology.

EasyLicense Backup and Restore

Apart from the EasyLicense type of license, it is not possible to back up or restore CrypKey licenses. The reason for this is that the other types of CrypKey licenses (those with time or run restrictions) are transferable by the user and the availability of backup-restore would allow the user to duplicate these licenses. However, the EasyLicense is not transferable and, consequently, it can be backed up and restored. You can think of EasyLicense as a permanent authorization of a program for a particular hard drive.

Procedures

It is simple to back up and restore a valid EasyLicense.

To back up EasyLicense:

Copy the following files to a safe place (floppy disk or another computer):

- `yourfilename.key`
- `yourfilename.rst`

To restore EasyLicense:

Install the software, if it is not already present, and simply copy the above two files back to the directory where they came from. If the software is on the same hard drive as it was when the license was backed up, the license is restored.

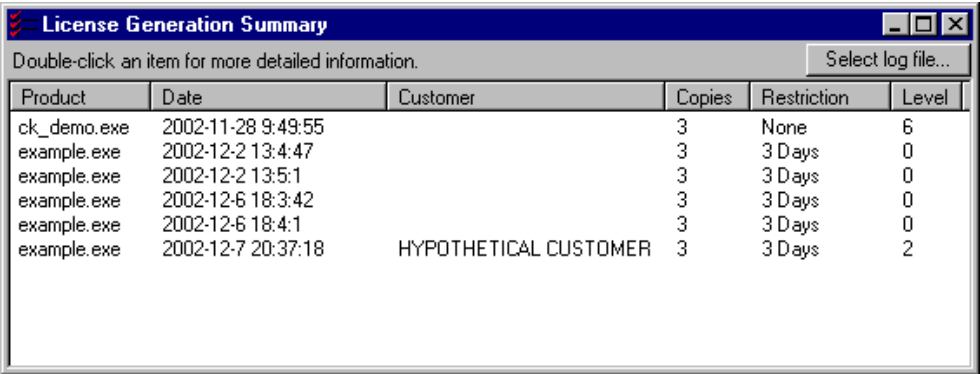
4.2.3 CloneBuster™ Technology

CrypKey 6.0, with the new patent-pending **CloneBuster™** technology, effectively solves the problem created by the upsurge in software products that clone hard drives. Many popular copy protection technologies that depend on placing some type of simple software footprint are now easily defeated by these products and are no longer an effective option. Hardware dongles provide limited protection against HD cloning, but are expensive and somewhat unpopular.

Most modern IDE hard drives have burned-in serial numbers (HDSN). The HDSN is a permanent read-only attribute of a personal computer hard drive, rather than the easily changed volume serial number. The HDSN is very difficult to access, but the CrypKey SDK 6.0 with CloneBuster™ technology has this ability. Utilizing CloneBuster™, CrypKey detects and logs the HDSN, model number, and firmware identifier and uses this information in its proprietary licensing algorithms. This ensures that your CrypKey-protected applications are not fraudulently transferred to unauthorized computers.

4.2.4 Transaction Summary

The Site Key Generator includes a **Summary** button that you can use to view license activity. When you click this button, the system displays the following window:



Product	Date	Customer	Copies	Restriction	Level
ck_demo.exe	2002-11-28 9:49:55		3	None	6
example.exe	2002-12-2 13:4:47		3	3 Days	0
example.exe	2002-12-2 13:5:1		3	3 Days	0
example.exe	2002-12-6 18:3:42		3	3 Days	0
example.exe	2002-12-6 18:4:1		3	3 Days	0
example.exe	2002-12-7 20:37:18	HYPOTHETICAL CUSTOMER	3	3 Days	2

Figure 30: Site Key Generator – License Generation Summary Window

4.3 Using the Site Key Generator

To start the Site Key Generator, you must first obtain the license to use. For details on obtaining the license, see Sec. 1.6.3: Obtaining Authorization for Your Site Key Generator.

Once you have obtained your Site Key Generator license, the main Site Key Generator window appears when you select the **SKW**  icon:

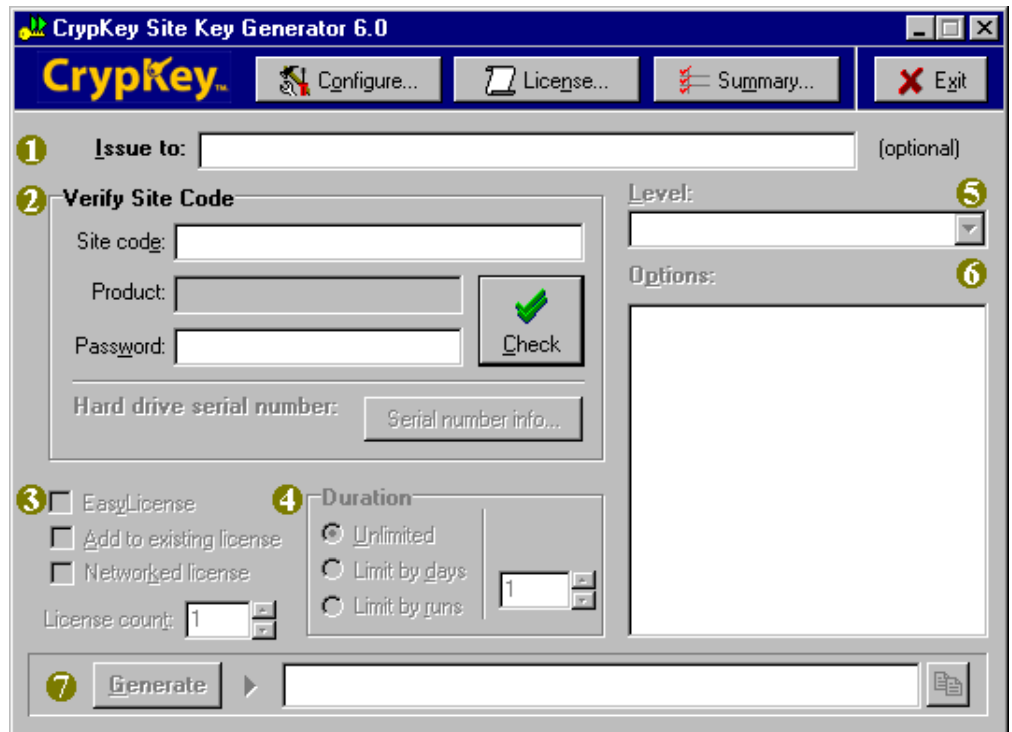



Figure 31: CrypKey Site Key Generator 6.0 Window

The Site Key Generator function handles the identification and general authorization of your clients, as well as the definition of your products and associated license terms, options, and levels.

4.3.1 Generating Site Keys

The CrypKey Site Key Generator 6.0 Window enables you to authorize your customers to use your product. When a customer loads your CrypKey-protected application on his or her computer, the application generates a site code. The customer sends you the site code, which you use in the Site Key Generator to produce a site key. You send this site key to the customer, who uses it to unlock your application under the terms you have defined in the Site Key Generator.

Procedure

1. Receive the customer's site code generated from your installed application.
2. Start the Site Key Generator, using the **SKW**  icon. The system displays the CrypKey Site Key Generator 6.0 Window.
3. Optionally, type in the customer name.

4. In the **Site code** field, type in the site code provided by the customer, then click the **Check** button. The window appears as follows:

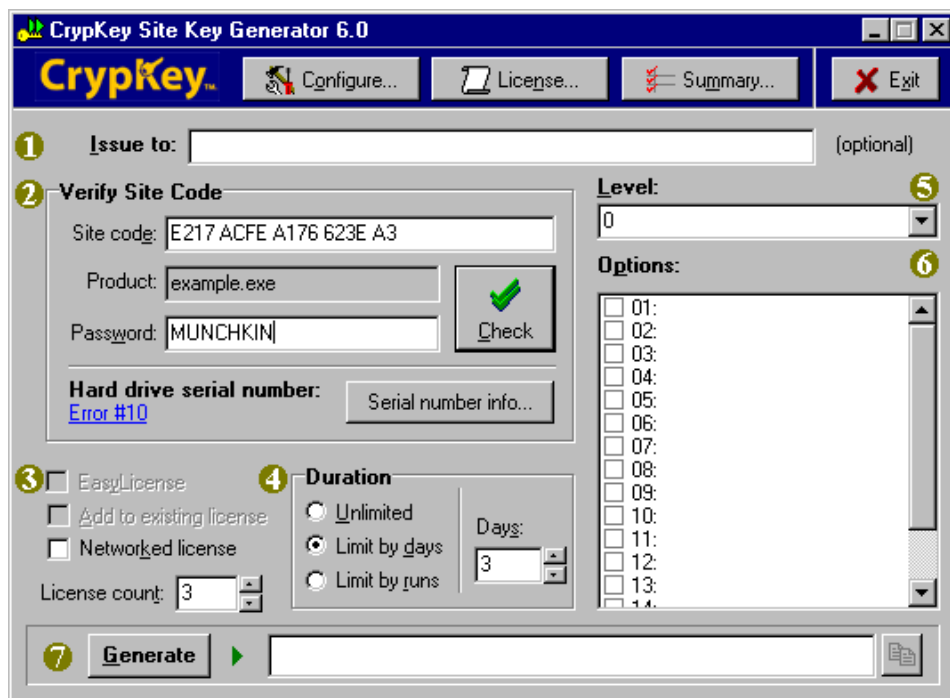


Figure 32: Site Key Generator – Starting Authorization

5. Click the **Serial number info** button to display the customer's hard drive serial number (HDSN), if available:

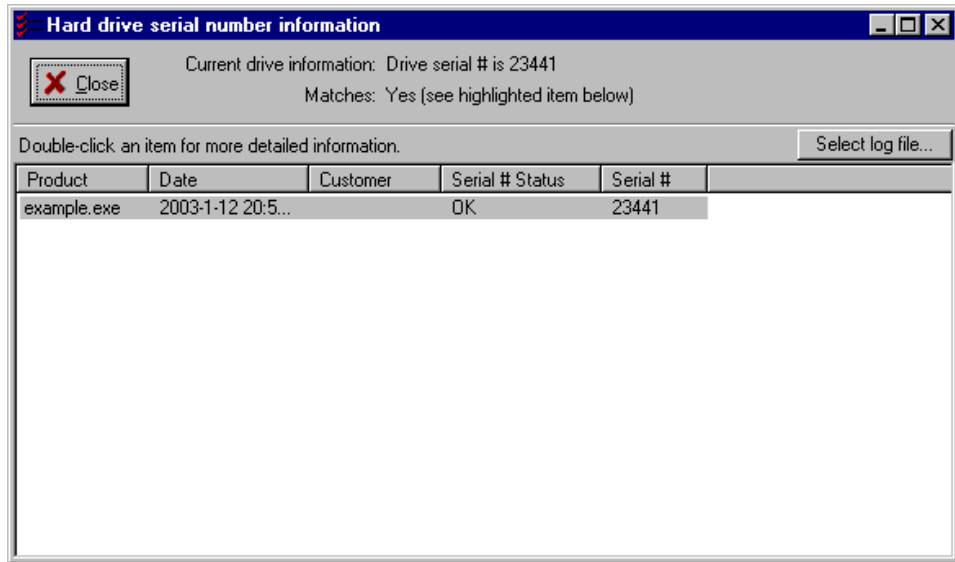


Figure 33: Hard Drive Serial Number Information Window

The HDSN exists on most computers. If CrypKey has a problem finding or reading the HDSN, the serial number is shown as unavailable in the HDSN Information window above.

You can double-click any information entry line in the above window, to display the License Details window shown below:

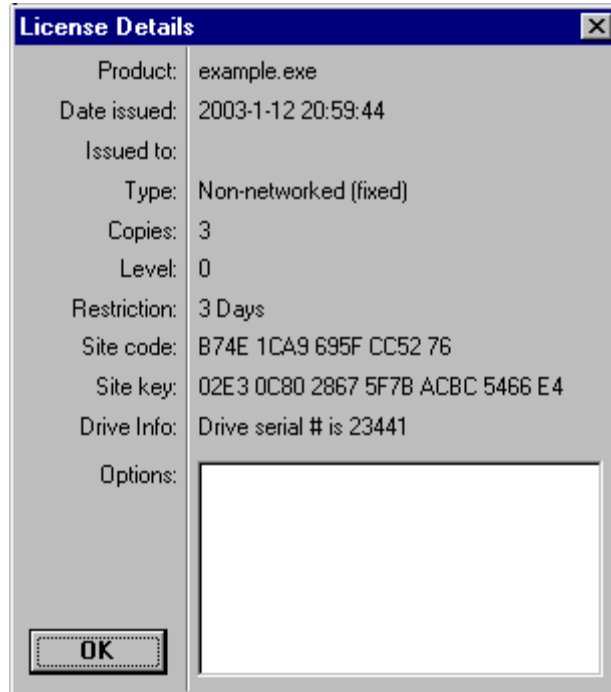


Figure 34: HDSN License Details Window

You should record the HDSN information for future reference. For example, it is helpful for verification purposes in the event that your customer asks you for re-authorization following a computer crash.

6. Verify the license terms generated from your application at your customer's computer. You can change the license duration, expressed as unlimited, days, or runs (for details, see the next section, Configuring Your Products).
7. In the **Level** box, click the drop-down arrow to assign one of the levels (if any) that were pre-defined during the product configuration.

8. In the **Options** box, click the checkboxes beside the product options that you wish to activate for the customer.
9. Click the **Generate** button.

The system generates a 26-character alphanumeric value and places it in the box beside the **Generate** button. This is the site key that you send to your customer. It contains all of the specifications you have entered concerning license terms, level, and options.

When your customer receives the site key and enters it into the CrypKey window of the installed application, the customer is authorized to use that application under the license terms defined.

4.3.2 Configuring Your Products

To specify the products that your clients is authorized to use, use the **Configure** window of your Site Key Generator. This is a separate operation from that of generating Site Keys.

Procedure

1. In the **CrypKey Site Key Generator 6.0** Window, click the **Configure** button.

The system displays the **Configure** window:

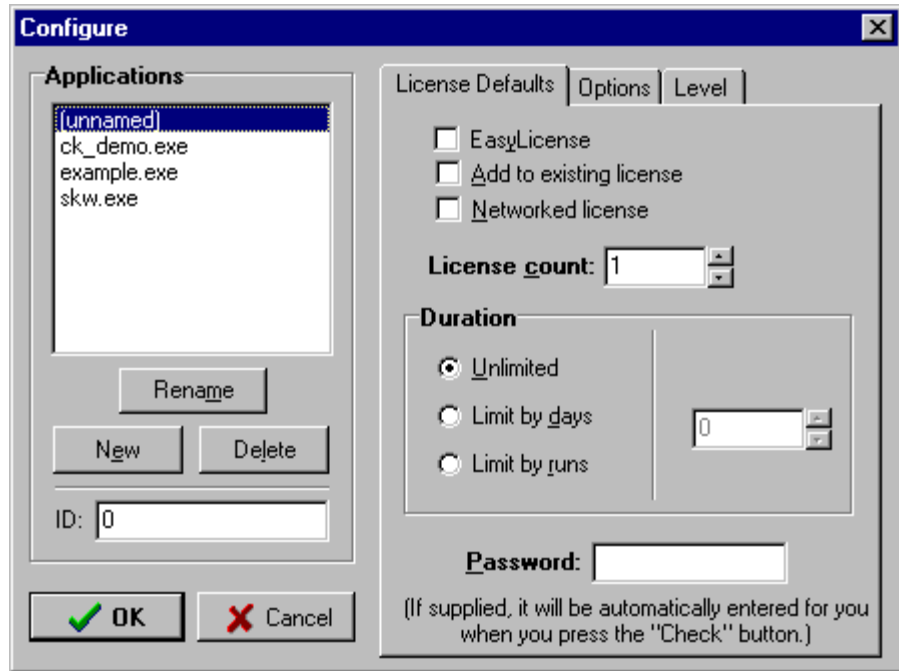


Figure 35: Configure Window

2. If you are editing an existing product file, click the applicable name in the **Applications** list and proceed to **Step 4**.

If you are configuring a new product, click **New** in the above window. The system displays the **Configure** window with the **New Application** popup superimposed onto it:

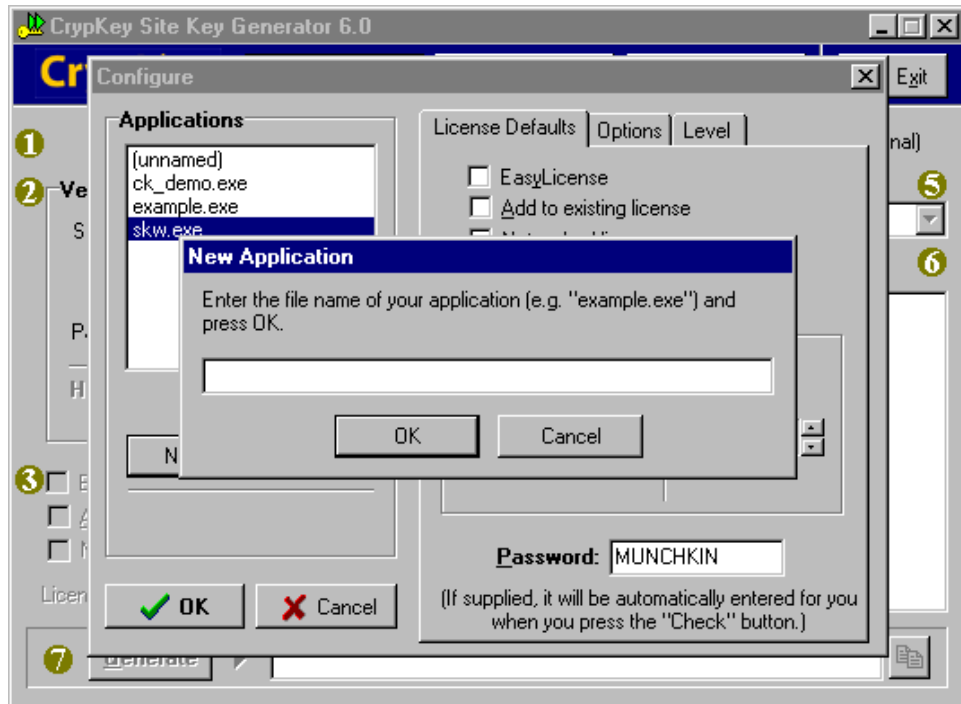


Figure 36: Site Key Generator with New Application Popup Window

3. Type the product's file name into the blank field and click **OK**. The popup window disappears and the name you have typed is displayed in the **Applications list**, as shown:

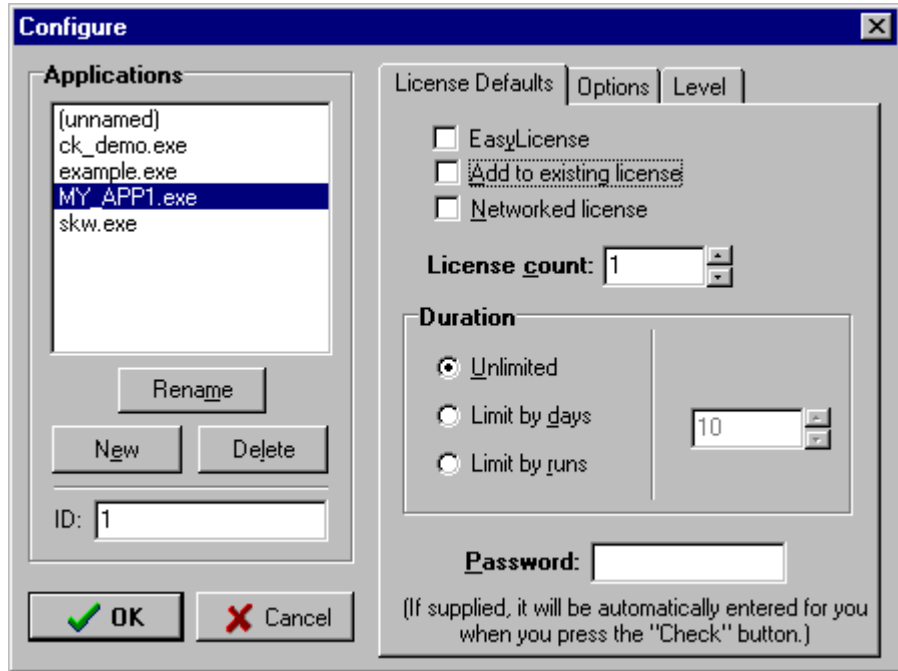


Figure 37: Configuration Window with New Application

You can rename or delete the new entry, or any existing entry, by clicking the name in the list, then clicking the **Rename** or **Delete** button.

4. In the **ID** field, enter a unique product identifier. For example, enter 1 if you are configuring your first product.
5. In the **License Defaults** tab page, set the default restrictions for the product. Note that the license automatically defaults to “Fixed” if you do not specify a network license.

The checkbox choices are as follows. These are defaults for the specified product and can be changed as needed for specific customers.

EasyLicense—unlimited license with no restrictions

Add to existing license—used if a customer with an existing license is specified in the CrypKey Site Key Generator 6.0 Window

Networked license—used if the application is to be run by one or more users connected to a server.

License count—maximum number of fixed or floating licenses that a customer can have for the specified product.

Duration—amount of usage allowed under the license. This can be unlimited (the default) or can be expressed in number of days or runs. If you select days or runs, the number box to the right becomes active and you are able to enter the number of days or runs allowed for the license.

Password—the password to be used by your customer. A password specified here is required in the **CrypKey Site Key Generator 6.0** Window when you are subsequently granting a license.

6. Click the **Options** tab to display the Options tab page:

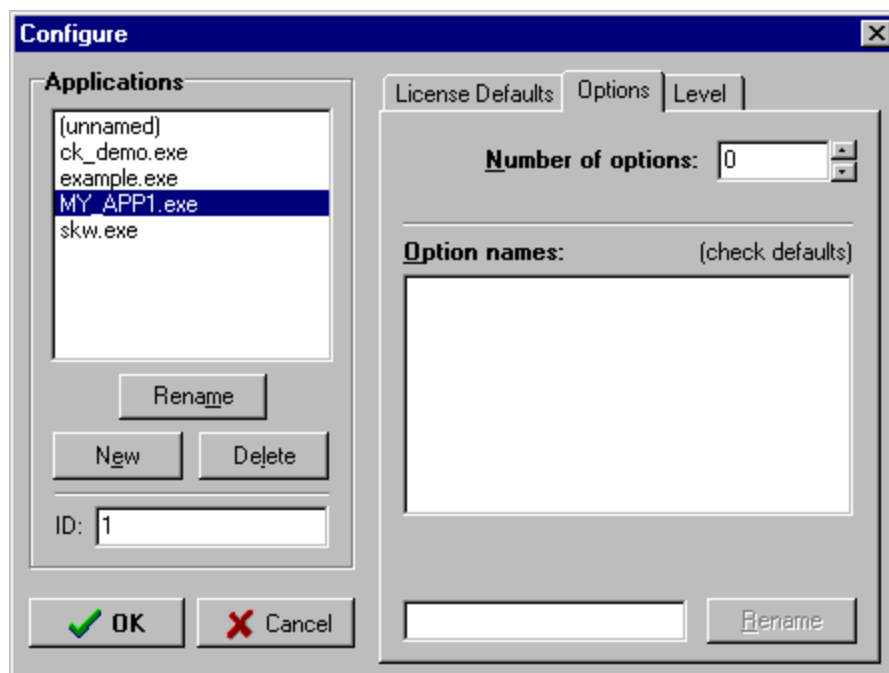


Figure 38: Site Key Generator – Options Tab Page

In the above window, click the combo box arrows to increase or decrease the number of options to be assigned to the product. As you increase the number, blank option entries appear in the **Option names** sub-window, as shown:

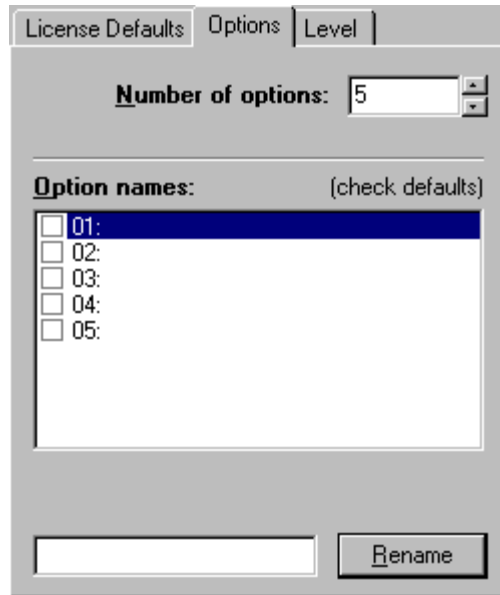


Figure 39: Site Key Generator – Options Combo Box

To assign a name to an option number, click the option's checkbox, type the name in the text box at the bottom of the window, and click the **Rename** button.

7. Click the checkboxes beside the options that you wish to assign to a license.

8. Click the **Level** tab to display the Level tab page:

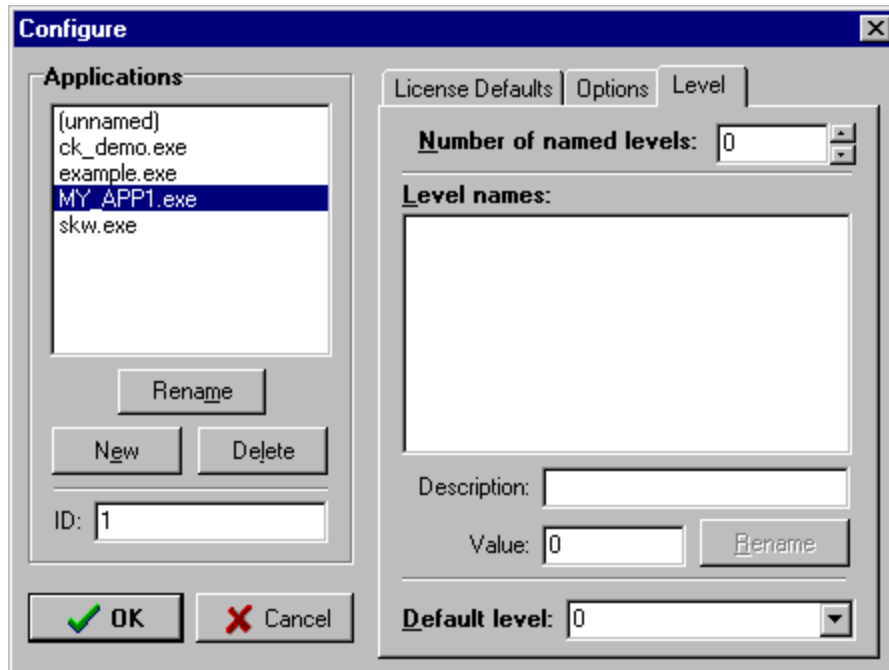


Figure 40: Site Key Generator – Levels Tab Page

In the above window, click the combo box arrows to increase or decrease the number of options to be assigned to the product. As you increase the number, blank option entries appear in the **Level names** sub-window, as shown:

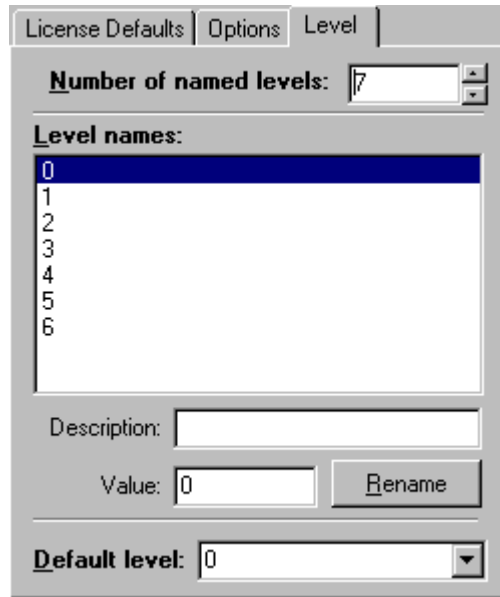


Figure 41: Site Key Generator – Levels Combo Box


To assign a name to a level, click on an entry number, type a name in the **Description** text box, and click the **Rename** button.

Note: only one level can be assigned per license.

4.3.3 Incrementing Existing Licenses

The Site Key Generator allows you to change the levels, options, type, and restriction of an existing client license. You can do this only if you have a valid site code from an authorized copy of CrypKey-protected software. You cannot toggle this option using your site code from the trial period

Procedure

1. Click the SKW  icon in the CrypKey program group to display the Site Key Generator main window.
2. Enter a valid site code into the **Site Code** field.
3. Click the **Add to Existing License** checkbox in the Site Key Generator dialog box, as shown:

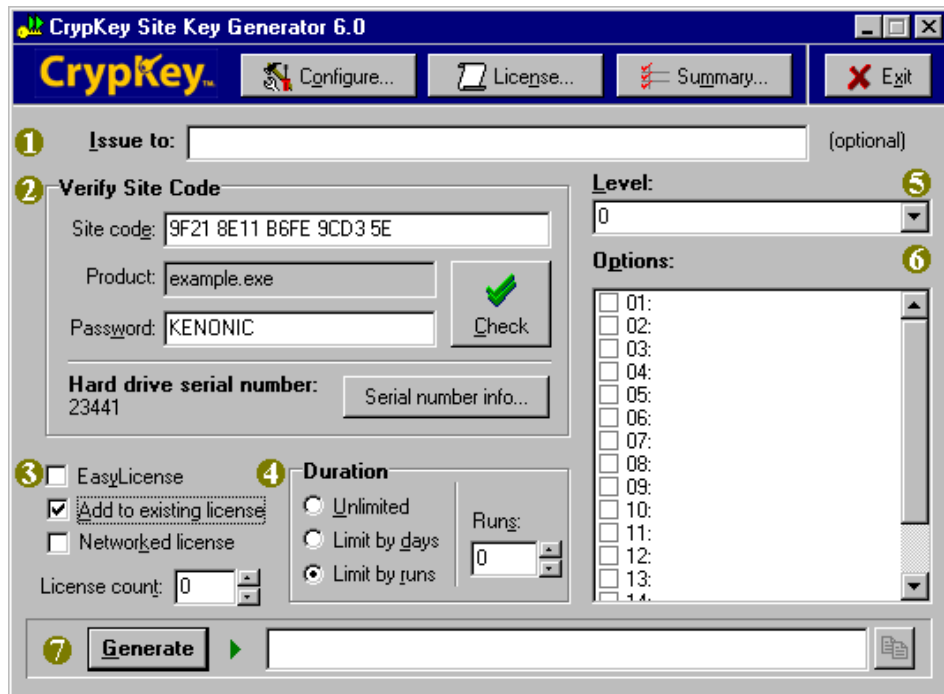


Figure 42: CrypKey Site Key Generator 6.0

4. Specify levels, options, and duration of the license. These changes are added to the license for the customer who provided the site code.
5. Click the **Generate** button to produce a new site key for the customer.

4.4 Distributor Authorizing License

The Distributor Authorizing License (DAL) allows you to authorize other Site Key Generators to authorize your product. You hold the master Site Key Generator and your distributor holds the slave Site Key Generator.

The benefit of bringing distributors into your business plan pertains to increased distribution and sales of your product. Distributors authorize **their** clients to run **your** product, based on the number of runs, days, or copies you specify, expanding your worldwide market base, hassle-free.

To obtain authorization for the DAL, send us the site key from your Site Key Generator once you have paid for this feature.

4.4.1 Authorizing Distributors

Your master Site Key Generator (SKG) can apply the following constraints to licenses assigned to a slave SKG:

- It can limit the number of licenses that the slave SKG can generate.
- It can limit the number of days during which the slave SKG can be used to distribute licenses.
- It can restrict the total limitation, in terms of days or runs, that the slave SKG can apply to licenses issued by it.

Note: A single slave SKG can issue only one type of license—either runs- or days-based, but not both. For example, if you want your distributor to issue 10 day- and 10 run-type licenses, then you must have them install copies of the `skw.exe` and `skw.ini` files into separate directories, licensing each slave SKG for a specific license type.

The number of licenses that a slave SKG can issue is based on the usage count. Each time the slave SKG generates a key, it decrements its usage count. The usage count of a slave SKG is regulated by the duration setting in the master SKG. The effects of various duration settings and the usage count are detailed in Table 2.

The license count should be set in the Master SKG to 1, unless you are issuing a network license for the SKG slave. The license count is the number of users who can simultaneously run the licensed application.

If your application has a non-network license, then the license count represents the number of licenses available to be transferred out. For example, if you license Product X as a non-network application with a license count of 3, the user could make a copy of the program on two other computers and transfer a license to each of them, making a total of three licensed copies.

Table 2: Effects of Duration Settings for Slave Site Key Generator

Duration Setting	Option Setting	Effect
unlimited		Slave SKG can generate an unlimited number of licenses for the specified product.
days = 10		Slave SGK can generate an unlimited number of licenses for the specified product, but stops functioning after 10 days.
runs = 15		Slave SGK can generate up to 15 licenses for the specified product.
unlimited	Days Only	Slave SGK can generate an unlimited number of day-type licenses (but not run-type licenses) for the specified product.
days = 10	Days Only	Slave SGK can generate an unlimited number of day-type licenses (but not run-type licenses) for the specified product, but stops functioning after 10 days.
runs = 15	Days Only	Slave SGK can generate up to 15 days of total license time. In this case, you could issue one 15-day license or three 5-day licenses for the specified product.
unlimited	Runs Only	Slave SGK can generate an unlimited number of run-type licenses (but not day-type licenses) for the specified product.

Duration Setting	Option Setting	Effect
days = 10	Runs Only	Slave SGK can generate an unlimited number of run-type licenses (but not day-type licenses) for the specified product, but stops functioning after 10 days.
runs = 15	Runs Only	Slave SGK can generate up to 15 runs of total license run time. In this case, the slave could issue one 15-run license or two 6-run and one 3-run license for the specific product.

Procedure

1. Send your distributor a copy of the Site Key Generator. This copy of the Site Key Generator is unauthorized at this stage.
2. Tell your distributor to:
 - a) Run the unauthorized copy of your Site Key Generator.

- b) Click **License** in the Site Key Generator License Authorization dialog box to display the following dialog box showing the site code:

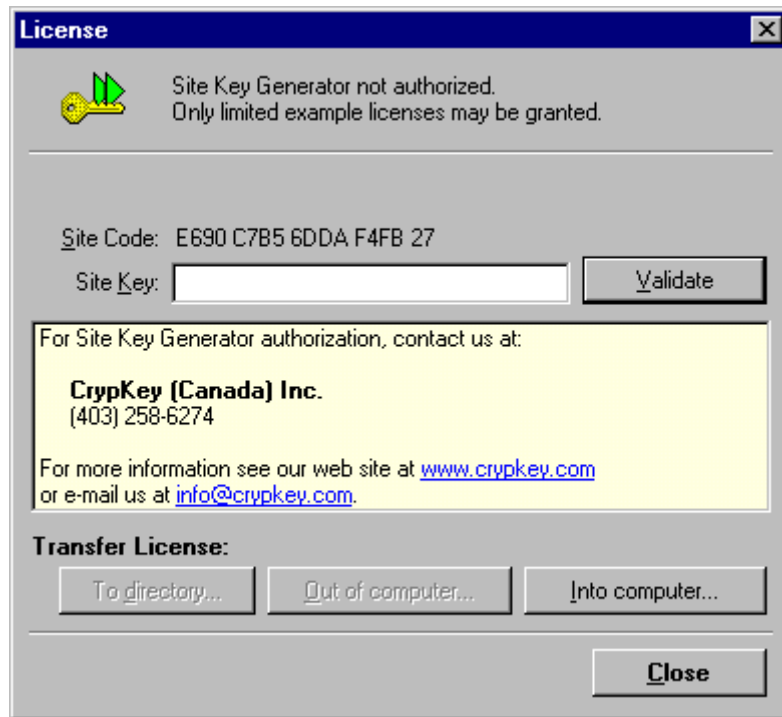


Figure 43: Distributor's Unauthorized SKG License Window with Site Code

- c) Provide you the site code (via email or telephone).
3. Type the site code from your distributor's slave Site Key Generator into the Site Code field in your master copy of the Site Key Generator.
 4. If the word MUNCHKIN doesn't already appear in the password field, type it in.

- Click the **Check** button in your Master SKG. After a few seconds, it appears as follows:

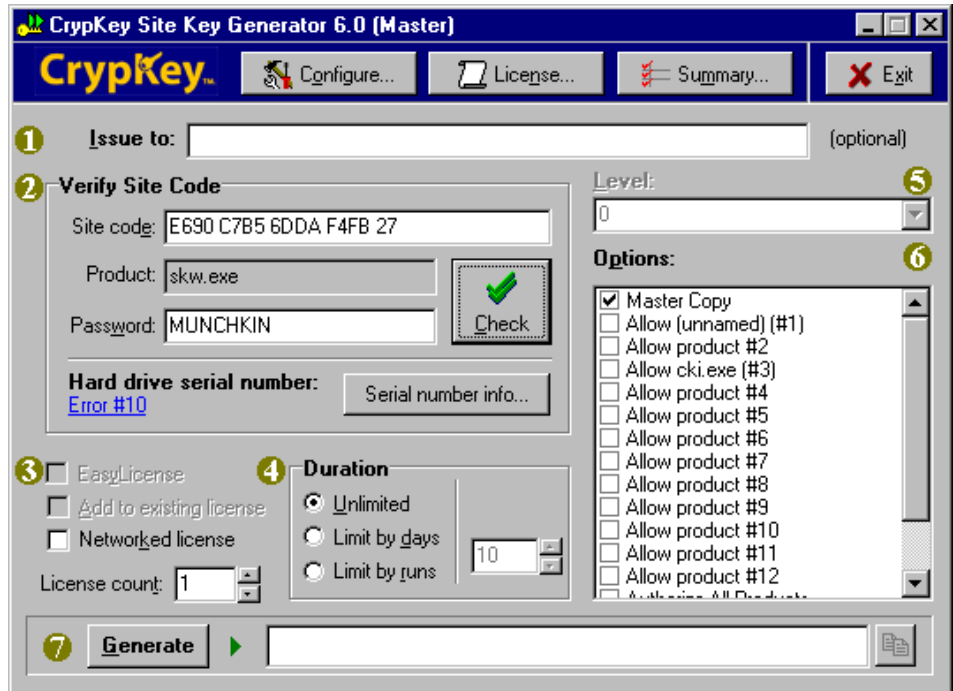


Figure 44: Master Site Key Generator with Distributor's Site Code

In the above window, the **Product** field should read `skw.exe`, which is the Site Key Generator executable file. The **Master Copy** option is selected by default in the **Options** panel.

- In the **Options** panel of the above window, select the products that you want to enable. You must enable at least one product, keeping in mind that the available options should already have been set up in the **Configure** tab page.

Note: when licensing a slave SKG, you should deselect the **Master Copy** option. If you leave it selected, you are authorizing a Master SKG, thereby allowing a distributor to create their own Site Key Generators with the capability of issuing licenses to distributors.

7. Click **Generate** to create the site key for your distributor. Your Master Site Key Generator now appears as follows (example selected options are shown):

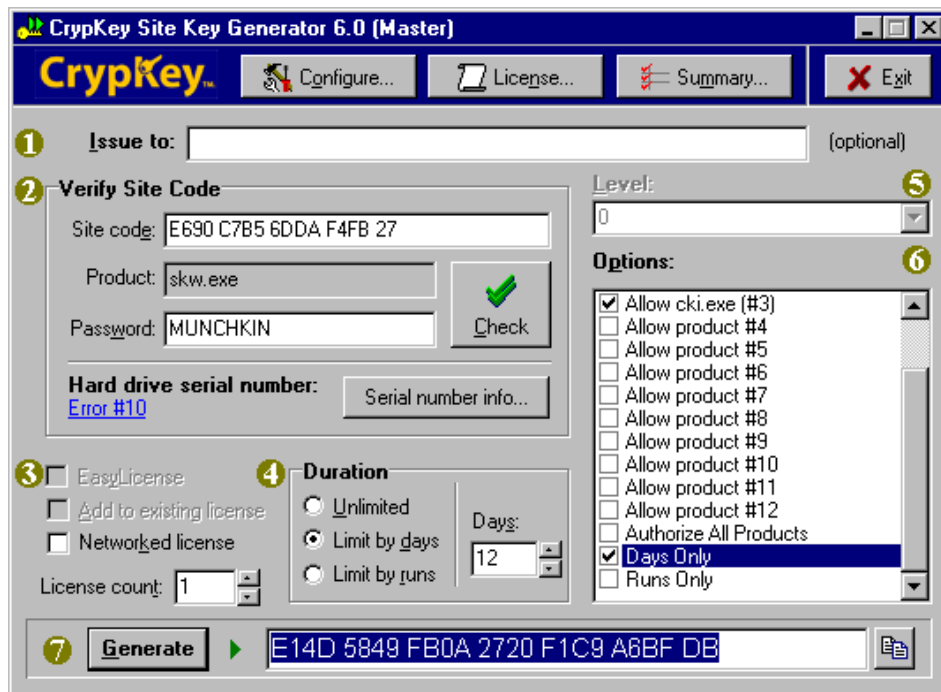


Figure 45: Master Site Key Generator with Slave Site Key and Selected Options

Note: the **Days Only** and **Runs Only** options, shown above, appear only when you authorize a slave Site Key Generator. They are used as special license restrictions for a slave SKG. When you enable **Runs Only**, and select the runs license, the slave Site Key Generator can only create runs licenses. In addition, the total number of runs the generator grants for a product is decremented from its own license count. Therefore, if a slave Site Key Generator is awarded a runs license with 1,000 runs, and the **Runs Only** option is enabled, the generator has 800 runs left after it has issued 200 runs to a product.

These rules apply in the same way to the **Days Only** option and days license.

8. Send the generated site key to your distributor.

9. Your distributor enters the site key into the License window of his or her Site Key Generator:

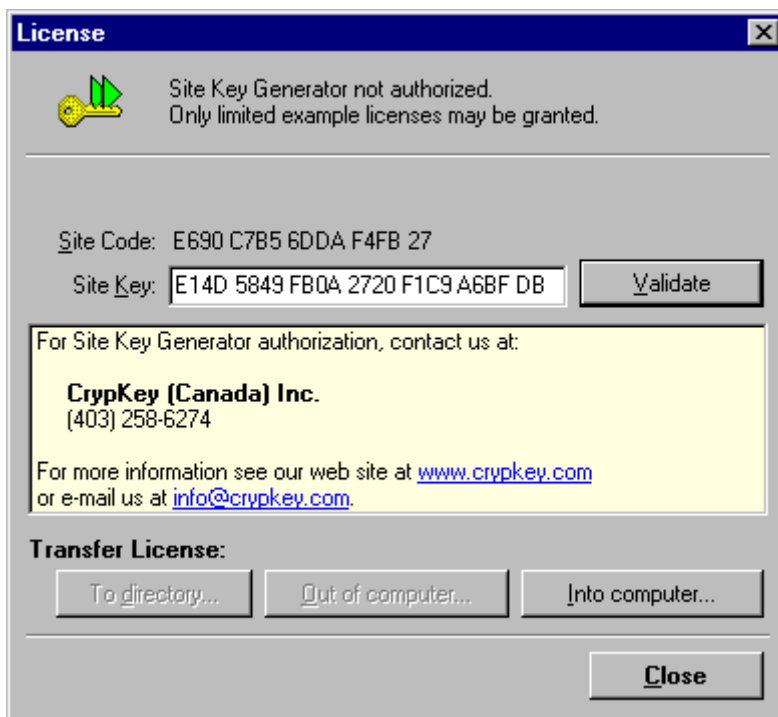


Figure 46: Site Key Generator License Authorization

10. Your distributor clicks the **Validate** button in the above window, resulting in authorization of the slave SKG as shown:

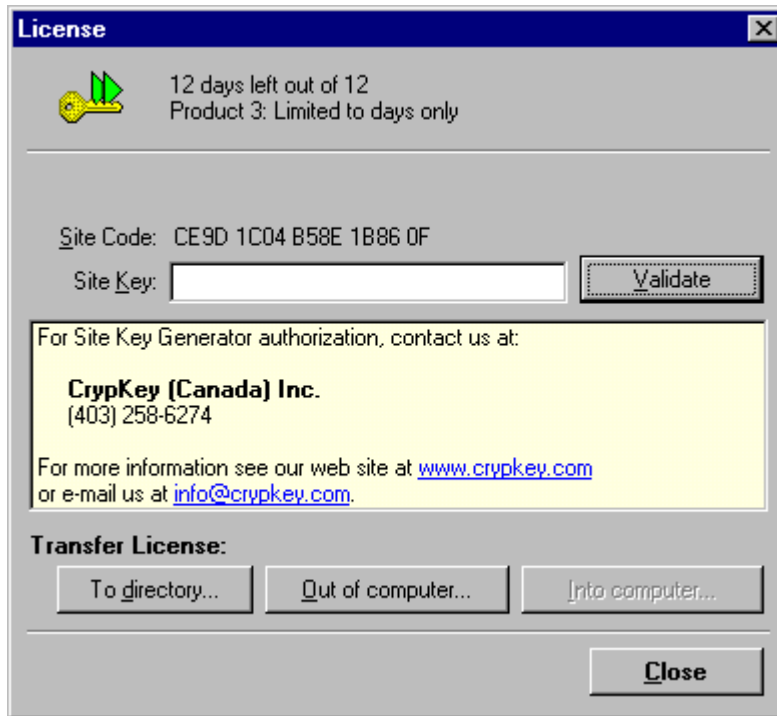


Figure 47: Slave SKG License Window – Authorized

Your distributor is now authorized to use the Site Key Generator to issue customer licenses for his or her products. Note that the license duration is the same as specified in the Master SKG (Figure 45).

You shouldn't need to contact your distributor again concerning issues of license validation, unless the distributor's hard drive crashes or he or she lose authorization.

Programming Your Application

The standard process for implementing CrypKey involves programming the SLAPI library functions into the source code of your application and providing the interface to your clients.

T

he compile and link procedure is the final stage of the process. See Section 13: Appendix I – Sample File for an example. There are many other pieces of sample code available in your CrypKey 6.0 sample directory.

5.1 Basic Programming Steps

The following steps show a suggested algorithm for establishing the license at startup. The steps are provided mainly to illustrate what the primary functions do. The steps you actually take may be very similar to the steps we describe, but your steps can be tailored for your application.

CrypKey SDK 6.0 operates very much as the previous version, with the exception that the **.ngn** file is required in your application directory. If you don't include this file, errors in the -200 range will likely result.

To establish the license:

1. Call the **InitCrypkey()** function.

This function initializes the CrypKey system with the path of the file to check for authorization. Call it on opening the application or starting the program, to initialize CrypKey.

This function returns 0 if successful (CrypKey initialized) and returns a negative value if unsuccessful (CrypKey not initialized). If the **InitCrypkey()** function returns a negative value, display an error message and disable or limit the program as desired. You may find it helpful to use the **ExplainErr()** function with the failure code to retrieve the appropriate error message).

Note that the **InitCrypkey()** and **GetAuthorization()** functions must be called at least once before any other CrypKey functions are called.

For details on the **InitCrypkey()** function, see the description in Section 12.3.26: **InitCrypkey()**.

2. Call the **GetAuthorization()** function.

The **GetAuthorization()** function checks whether or not the program is authorized to run and returns the level/options information if the program is authorized. The function is also used to increment the count of runs used.

Call this function on opening the application or starting the program, to check whether or not the user is authorized to run the program, and call it whenever you need to check the level or option setting for the user's license. Call this function as often as possible in order to strengthen the security of your application even further.

This function returns 0 if successful (program authorized), a negative value if unsuccessful (program not authorized), or a positive value (signifying that the program has a multi-user license and more users are requesting use of the program than the license allows).

If the **GetAuthorization()** function returns 0, proceed to step 3 only if the user wants to change the license permissions.

If the **GetAuthorization()** function returns a negative value, display an error message informing the user that the program is not authorized, optionally with an

explanation (you may find it helpful to use the **ExplainErr()** function with the failure code to retrieve the appropriate error message). Continue as outlined in step 3 to prompt the user to authorize the program, possibly at a later time (i.e., off a menu item command).

If the **GetAuthorization()** function returns a positive value, which is a queue value indicating the number of users who must quit the application before a license becomes available, continue repeating calls to **GetAuthorization()** until the user gets a license or gets tired of waiting and quits.

Note that the **InitCrypkey()** and **GetAuthorization()** functions must be called at least once before any other CrypKey functions are called, and **GetAuthorization()** must also be called repeatedly at regular intervals when network licensing is in effect.

See the **GetAuthorization()** function description in Section 12.3.13: **GetAuthorization()** for more details.

3. Call the **GetSiteCode()** function.

This function gets the site code for the program. Call it when you want to authorize a new license or change the current license permissions. It returns 0 if successful, meaning that the site code is returned, or returns -1 if CrypKey is not yet initialized.

You must display the site code to the user in order for the user to be able to provide it to you when he or she calls you to request a site key (for instructions on creating the site key, see Section 4: Site Key Generator. The user can shut down the program while waiting for the site key; the site code is not changed when the program starts up again.

See the **GetSiteCode()** function description in Section 12.3.24: **GetSiteCode()** for more details.

4. Call the **SaveSiteKey()** function.

The **SaveSiteKey()** function saves the site key information, which contains your product's licensing permissions for the customer's site. Encrypted license files are created from this function and saved on the computer's hard disk. Call the function after the user has entered the site key.

If the **SaveSiteKey()** function returns 0, a new license is created for the customer's site. If the **SaveSiteKey()** function returns a negative value, notify the user that the site key is invalid. You may find it helpful to use the **ExplainErr()** function with the failure code to retrieve the appropriate error message.

Note that if the site key is saved successfully, the site code changes. This is done to prevent the user from reusing an old site key. If you give the user a site key that allows 10 runs, you don't want the user to reuse that old key later for another 10 runs, even if recycling is good for the environment.

See the **SaveSiteKey()** function description in Section 12.3.32: **SaveSiteKey()** for more details.

5. Call the **EndCrypkey()** function.

The **EndCrypkey()** function ensures that the licensing control system terminates cleanly without errors. Call this function just before exiting the program. There is no return value.

Note that if **EndCrypkey()** is not called before the program terminates, various errors may be encountered the next time the program is started and the **InitCrypkey()** function is called.

See the **EndCrypkey()** function description in Section 12.3.7: **EndCrypkey()** for more details.

5.2 GUI Programming Tasks

In addition to checking to see if the program is authorized when your application is started, you must provide menu options that enable the user to authorize your product and perform authorization transfers.

5.2.1 Authorizing the Program

Procedure

To authorize the program:

1. Execute the **GetSiteCode()** function.
2. Check the return value and code the appropriate result (i.e., disable the program).
3. Open a window passing the site code obtained from the **GetSiteCode()** function to the user.
4. Enter the site key in the window and presses an **OK** button.
5. Execute the **SaveSiteKey()** function.
6. Check the return value and code the appropriate result (i.e., disable the program).

5.2.2 Registering the Authorization Transfer on the Target Computer

Procedure

1. Open a window requesting the directory path.
2. The user enters or selects the default path in the window and presses an **OK** button.
3. Execute the **RegisterTransfer()** function.
4. Check the return value and code the appropriate result (i.e., disable the program).

5.2.3 Transferring the Authorization Out of the Source Computer

Procedure

1. Open a window requesting the directory path.
2. The user enters or selects the default path in the window and presses an **OK** button.
3. Execute the **TransferIn()** function.
4. Check the return value and code the appropriate result (i.e., disable the program).

5.2.4 Transferring the Authorization into the Target Computer

Procedure

1. Open a window requesting the directory path.
2. The user enters or selects the default path in the window and presses an **OK** button.
3. Execute the **TransferOut()** function.

5.3 Compiling and Linking

The following sections explain how to compile and link CrypKey into your C or C++ application.

5.3.1 CrypKey Header File

Procedure

Include `crypkey.h` in all modules that make calls to the CrypKey library. If you are using the `CRP32DLL.DLL` with a C or C++ module, you must include the following statement:

```
#define DLL
```

5.3.2 Linking

Link the appropriate library to your application, depending on the compiler and platform. All libraries have been compiled using the Large memory model. CrypKey requires ample stack space—you should allow at least 10K.

The table below describes the available libraries. See also the notes following the table.

Table 3: Libraries

Library	Compiler	Platform	Type
CRYPKEYCOM.DLL	All compilers that can use COM objects	Win32	COM Object
CrypKeyNET.DLL	All compilers that can use .NET objects	.NET Framework	.NET assembly
LCRYPKB.LIB	Borland	MS-DOS 16-bit	Static library
LCRYPKY7.LIB	Microsoft C 7.0	MS-DOS 16-bit	Static library
LCRYPKYM.LIB	Microsoft C 7.0/VC++	Win16	Static library
LCRYPKYD.LIB	Microsoft C 7.0/VC++	Win16	Import library for DLL
LCRYPKYD.DLL	Any Win16-based compiler	Win16	Dynamic link library
LCRYPKYB.LIB	Borland C++	Win16	Static library
CRP32S42.LIB	Microsoft VC++ up to 5.x	Win32	Single-thread static RT library
CRP32D42.LIB	Microsoft VC++ up to 5.x	Win32	Multi-thread dynamic C RT library
CRP32M42.LIB	Microsoft VC++ 4.2	Win32	Multi-thread static RT library
CRP32D60.LIB	Microsoft VC++ 6.0	Win32	Multi-thread dynamic RT library
CRP32M60.LIB	Microsoft VC++ 6.0	Win32	Multi-thread static RT library

Library	Compiler	Platform	Type
CRP32S60.LIB	Microsoft VC++ 6.0	Win32	Single-thread static RT library
CRP32DLL.DLL	Any Win32-based compiler	Win32	Multi-thread RT library
CRP32DLL.LIB	Microsoft VC++ 4.2	Win32	Import library for CRP32DLL.DLL
CRP32BST.LIB	Borland C++ 4.51	Win32	Single-thread static library
CRP32BMT.LIB	Borland C++ 4.51	Win32	Multi-thread static library
CK16RMV.EXE	Any 32-bit program	Win32	Win 95/98/ME libraries
CRYP95F.DLL	Any 32-bit program	Win32	Win95/98/ME libraries
CRP9516F.DLL	Any 32-bit program	Win32	Win95/98/ME libraries
CRP32001.NGN	Any 32-bit program on Windows NT, 2000 or XP	Win32	Same directory as 32-bit application or CK license

Note: If you are using Microsoft VC++ in a Win32 application that dynamically links to Windows C runtime library, you **must** use CRP32D42.LIB or CRP32D60.LIB; otherwise, you link errors will result.

CrypKey can also be used within non-C applications by applying the dynamic link library (DLL).

Library Notes

If you are getting unresolved externals or multiple defined externals, it's probable that you are linking to the wrong library or have a model combination that we do not support.

Microsoft Visual C (MSVC) allows:

- single or multiple threads
- static or dynamically linked runtime libraries

There is a different C runtime library for each combination of these pairs and the compiler automatically links to the correct one for you.

You must also make a different CrypKey library for each combination and tell the compiler the correct one to link to.

Most MSVC sample projects default to multithread and dynamic runtime C Lib (CRT), which is a combination we don't support. We recommend the combination of multithread with static libraries, as fewer problems occur with this model.

CrypKey uses three of the possible four model combinations. The following table shows the library names and the combinations for which they are used:

Table 4: Library Linkages

Name	Type	Linkage
CRP32D60.LIB	Multi thread	Dynamic CRT
CRP32M60.LIB	Multi thread	Static
CRP32S60.LIB	Single thread	Static

To use development tools such as Visual Basic, Access, Delphi, and PowerBuilder with CrypKey:

1. Ensure that your application is able to make calls to dynamic link libraries. Refer to development system's manual for instructions on how to call and link DLL functions.
2. Declare/define your external CrypKey functions within your application.
3. Since you are unable to use the crypkey.h library file, which contains error handling variable declarations, you may use the integer value provided within your programming or declare your own variables within your application.
4. Ensure that you have installed the lcrypyd.dll or crp32dll.dll library file. You must include this file with your installation disks.
5. Call the **CKChallenge()** function to ensure that the file CRYP32DLL.DLL has not been replaced with an impostor. This function requires two additional secret numbers that are provided with your developer keys. You can use the numbers in example.c for testing purposes.

The secret numbers are the password number and the company number you receive with your master and user keys. Following is an example set of these numbers.

The Master Key for EXAMPLE.EXE, company 7956123, product 1, Floating:Y, NT:Y, NT_32_Bit_Lib:Y, is:

2A5D 57C4 1B4C 135B F09E 17F7 600B 2D70 79E8 F275 C32A

The User Key for company 7956123 is:

D050 815C D1A2 A79D B103

The password number is:

984534120

5.4 COM Object

The CrypKey COM Objects are tools that have proven useful in configuring protection for applications and issuing licenses. The essential information you need on this topic is provided here. Additional detail is available in the document named *CrypKeySDK COM Object*, which are found in your CrypKey directories.

Using Dynamic Link Libraries (DLLs) with Visual Basic and other languages has always been a difficult task. Since LIBs are encapsulated in a COM object, the incorporation of CrypKey SDK functions into an application becomes much easier. This is particularly true with Visual Basic (VB), where COM support is built in. COM objects are also supported by C++, Java, and web development tools like IIS (InterDev) and Cold Fusion.

Note: .NET assemblies are similar to COM objects and are used in essentially the same way. However, .NET assemblies are available only to .NET languages (VB.NET, C#.NET, C++.NET).

For each CrypKey function, you will find the prototypes used in C, C++, VB and C#.NET in Sec. 12.3: Function Descriptions.

- The C prototype is for use with the CrypKey Library.
- The C++ and VB prototypes are for use with the CrypKey COM Object.
- The C#.NET prototype is for use with the CrypKey .NET assembly.

5.4.1 COM Object Identifiers

The COM objects in the library have specific names and types, as shown:

Table 5: COM Object Identifiers

Object	Name	C++ Type
SDK	CrypKey.SDK	ICrypKeySDKPtr
Float Record	CrypKey.SDKFloatRecord	ICrypKeySDKFloatRecordPtr

5.4.2 Installing a COM Object

The following procedure assumes that you have completed the CrypKey base product installation on the system you are using.

Procedure

1. Switch to the directory into which you are installing the files.
2. Execute the following command:

```
regsvr32 CrypKeyCOM.dll
```

You are now ready to use the COM object from anywhere in the system. You do not have to copy the COM object file or register it more than once.

There are two methods of using the COM object with Visual Basic:

- referencing the COM Object explicitly
- referencing the COM Object by name

The above methods are described in the sections that follow.

A third method, also described below, uses the COM object in C++.

5.4.3 Referencing a COM Object Explicitly

To use this method, start Visual Basic and choose the menu Project/References. In the **References** window, find the item named CrypKey COM 1.0 Type Library and ensure there is a check beside it. Press **OK** to close the window. Now you can write your code like this:

```
Dim myObject As New CrypKeySDK
Dim initResult As Long

' initialize CrypKey
initResult =
myObject.InitCrypKey("c:\my_directory\my_product.exe", _
    "<<replace this with your master key>>", _
    "<<replace this with your user key>>", 0, 300)

' check if call successful
If (initResult = 0) Then
    ' check if we are authorized
    Dim authResult As Long
    Dim nOptLevel As Long
    authResult = myObject.GetAuthorization(nOptLevel, 1)

    ' check if call successful
    If (authResult = 0) Then
        MsgBox "You are authorized with these options/
        levels" & nOptLevel
        ' you should run your program now
    Else
        MsgBox "You are not authorized to run this
        application!"
        ' you should abort your program now
    End If
Else
    MsgBox "Initialization failed!"
    ' you should abort your program now
End If
```

5.4.4 Referencing a COM Object by Name

To use this method, no special project setup is needed. You can write your code exactly as shown in Sec. 5.4.3, except that the first two lines are replaced with the following four lines:

```
Dim myObject As Object
Dim initResult As Long

' create an instance of the CrypKey COM object
Set myObject = CreateObject("CrypKey.SDK")
```

The rest of the coding, starting with CrypKey initialization, is similar to that shown in Sec. 5.4.3, with minor modifications.

5.4.5 Pre-defining DLL functions

To illustrate the ease of using the COM object, for matters of comparison, the old method of pre-defining DLL functions is described here.

To use this method, you must define each function in the CrypKey SDK DLL that you call and ensure that the DLL is in the application or system path. This method is more confusing and prone to errors than using a COM object. It looks something like this in VB:

```
' EXTERNAL FUNCTION DECLARATIONS

' Crypkey Functions
Declare Function InitCrypkey& Lib "crp32dll.dll"
    (ByVal filepath$, ByVal MasterKey$, ByVal UserKey$,
    ByVal allow_floppy&, ByVal network_max_checktime&)
Declare Function SaveSiteKey& Lib "crp32dll.dll"
    (ByVal site_key$)
Declare Function GetSiteCode& Lib "crp32dll.dll"
    (ByVal site_code$)
Declare Function GetAuthorization2& Lib "crp32dll.dll"
    (ByVal decrement&)
Declare Function GetAuthorization& Lib "crp32dll.dll"
    (ByRef oplevel&, ByVal decrement&)
```

```

Declare Function Get1RestInfo& Lib "crp32dll.dll"
    (ByVal which&)
Declare Function readyToTryDays& Lib "crp32dll.dll"
    (ByVal oplevel&, ByVal numdays&, ByVal version&, ByVal
copies&)
Declare Function GetLevel& Lib "crp32dll.dll"
    (ByVal DefinedLevels&)
Declare Function GetOption& Lib "crp32dll.dll"
    (ByVal nDefinedOptions&, ByVal nOptionNum&)

```

The rest of the coding, starting with CrypKey initialization, is similar to that shown in Sec. 5.4.3, with minor modifications, using a COM Object in C++.

5.4.6 Using a COM object in C++

You must reference the object DLL in your source code. This picks up the type library for the object and allows the compiler to compile your application. Following is a sample of the source code for this.

```

#import "CrypKeyCOM.dll" no_namespace named_guids
void main ()
{
    // initialize COM library
    CoInitialize (NULL);

    {
        // create an instance of the COM object
        ICrypKeySDKPtr ptr;
        ptr.CreateInstance ("CrypKey.SDK");

        // initialize CrypKey properly
        long nResult = ptr->InitCrypKey (_T("c:\\example.exe"),
            T("2A5D 57C4 1B4C 135B F09E 17F7 600B 2D70 79E8
            F275 C36A"),
            T("D050 815C D1A2 A79D B103"), 0, 300);

        // check if call successful
        if (nResult == 0)
        {
            // check if we are authorized

```

```

        long nAuthResult;
        long nOptLevel;
        nAuthResult = ptr->GetAuthorization (&nOptLevel,
        1);

        // check if call successful
        if (nAuthResult == 0)
        {
            printf ("You are authorized with these
            options/levels %d",
            nOptLevel);

            // you should run your program now
        }
        else
        {
            printf ("You are not authorized to run this
            program!");

            // you should abort your program now
        }
    }
else
{
    printf ("Initialization failed!");
}
}
}

```

5.4.7 COM Object Constants

The CrypKey SDK COM Objects use the data types described below for internal operation.

Data type: *CKExplainEnum*

Error Name	Value	groups of error codes to be used with the ExplainErr methods
CKAuthError	1	Authorization methods errors
CKSiteCodeError	2	Get Site Code methods errors
CKSaveSiteKeyError	3	SaveSiteKey errors
CKRegisterTransferError	4	RegisterTransfer errors
CKTransferOutError	5	TransferOut errors
CKTransferInError	6	TransferIn errors
CKDirectTransferError	7	DirectTransfer errors
CKInitializeError	8	InitCrypKey errors
CKReadyToTryError	9	Ready To Try methods errors
CKKillLicenseError	10	KillLicense errors
CKAcquireError	11	Acquire methods errors
CK HardDriveError	12	Hard Drive Serial Number methods errors
CKFloatSnapError	13	Floating License Snapshot methods errors
CKCrypKeyCOMInternal	999	Errors internal to this COM object

Data type: *CKRestrictionTypeEnum*

Condition Name	Value	License types from GetRestrictionInfo
----------------	-------	---------------------------------------

CKRestrictionTypeNone	0	license with no restrictions
CKRestrictionTypeTime	1	license with a time (number of days) restriction
CKRestrictionTypeRuns	2	license with a runs (number of runs) restriction

Data type: *CKBooleanValueEnum*

Condition Name	Value
----------------	-------

CKBooleanFalse	0
CKBooleanTrue	1

Data type: *CKMaxLengthEnum*

Parameter	Value
-----------	-------

CKMaxLenComputerName	15
CKMaxLenUserName	15

5.4.8 Internal COM Status Messages

Table 6: Internal COM Status Messages

Message	Value	Meaning
CKInternalOk	-1000	Operation is normal.
CKInternalNullPointer	-1001	A NULL pointer was passed as a parameter.

Message	Value	Meaning
CKInternalMax	-1002	Flags this value as the last value in the CKInternal table of values.

5.4.9 Referencing the Floating License Snapshot Record

The COM object contains functions used to encapsulate the Floating License Snapshot (FLS) record. The following table contains C++ and C#.NET prototypes for getting or setting each data item in the FLS record. The Item Names in this table parallel the names in the FLS record itself (see Sec. 12.3.11: FloatingLicenseSnapshot).

Table 7: COM C++ and C#.NET Prototypes for FLS Record Data Items

Item Name	Language	Prototype	Description
nId	C++	HRESULT get_nId (long *pVal);	Gets the unique internal ID.
		HRESULT put_nId (long newVal)	Sets the unique internal ID.
	C#.NET	property int nId	Gets/Sets the unique internal ID.
nUpdate	C++	HRESULT get_nUpdate (long *pVal);	Gets the internal timestamp of the last update.
		HRESULT put_nUpdate (long newVal)	Sets the internal timestamp of the last update.
	C#.NET	property int nUpdate	Gets/Sets the internal timestamp of the last update.
nStatus	C++	HRESULT get_nStatus (long *pVal);	Gets the status of the license; 0=running, positive number=queue order.

Item Name	Language	Prototype	Description
		HRESULT put_nStatus (long newVal)	Sets the status of the license; 0=running, positive number=queue order.
	C#.NET	property int nStatus	Gets/Sets the status of the license; 0=running, positive number=queue order.
Ntimest amp	C++	HRESULT get_nTimestamp (long *pVal);	Gets the timestamp of when the license was started; number of seconds since 1900.
		HRESULT put_nTimestamp (long newVal)	Sets the timestamp of when the license was started; number of seconds since 1900.
	C#.NET	property int nTimestamp	Gets/Sets the timestamp of when the license was started; number of seconds since 1900.
StrUser Name	C++	HRESULT get_strUserName (BSTR *pVal);	Gets the user name.
		HRESULT put_strUserName (BSTR newVal)	Sets the user name.
	C#.NET	property string strUserName	Gets/Sets the user name.
StrComp uterName	C++	HRESULT get_strComputerName (BSTR *pVal);	Gets the computer name.
		HRESULT put_strComputerName (BSTR newVal)	Sets the computer name.
	C#.NET	property string strComputerName	Gets/Sets the computer name.

Item Name	Language	Prototype	Description
strSpare	C++	HRESULT get_strSpare (BSTR *pVal);	Gets the data for internal use only.
		HRESULT put_strSpare (BSTR newVal);	Sets the data for internal use only.
	C#.NET	property string strSpare	Gets/Sets the data for internal use only.

5.5 External Linking using Binary Files

CrypKey can be externally linked to your program using files to transfer information between your program and a CrypKey User Interface executable (cui.exe) called by your program. The cui.exe is supplied by CrypKey and is in the CrypKey.60\Demos\16Bit directory. It is a simple MS-DOS program that makes all calls to the CrypKey C library and provides the user with an interface to all CrypKey licensing functions.

Rather than making the CrypKey calls yourself, your program need only write a request file, call cui.exe, and read the reply file left by cui.exe. The advantages of using this type of link are:

- Your program is smaller since the CrypKey library is not linked to it.
- You do not need to program the CrypKey interface, since the CrypKey User Interface executable handles this for you.
- You can use CrypKey from programs in any language or compiler that can write a binary file.

5.5.1 External Linking Security

CrypKey external linking is designed to be a secure process. The following steps have been provided to ensure a secure, tamper-proof link between your program and cui.exe:

- The files used to transfer the information are encrypted. The encryption key is a number that you choose at random to customize the encryption for your company. The encryption key is located in cui.h. You should change this to a random number of your choosing and recompile cui.exe. This customizes cui.exe for your company.

- Your program issues a unique random number in the request file that must be echoed in the reply file. This ensures that the reply file is always custom made for each request. These same steps are used to secure the site code/site key transaction and have proved highly effective.

5.5.2 Transferred Information

The following table describes the information via external links.

Table 8: Transferred Information Descriptions

Name	Size (Bytes)	Range	Description
RandomNum	2	0-32768	This is used to secure the information exchange. Use this number in the CKChallenge() calculation and compare your results to the results passed back by cui2.exe.
RandomTwo	2	0-32768	This is also used to secure the information exchange, along with RandomOne.
Title	Variable	Any char	This is a text title displayed by cui2.exe to your customer.
Filepath	Variable	Any char	The path of the protected executable file.
UserKey	Variable	Any char	Issued by CrypKey.
MasterKey	Variable	Any char	Issued by CrypKey.
Decrement	2	0-65536	The number of runs that you want to decrement from the total.
Allow Floppy	2	01	Set this to 0.
NetHandle	2	0-65536	This is used to query an already open network license. If you are not allowing network usage of your program, set this to 0.

Name	Size (Bytes)	Range	Description
NetworkMax CheckTime	2	0-65536	This is the time in seconds after which a network copy is deemed to be hung if it has not checked its authorization. See the InitCrypkey() function description in Section 12.3.26: InitCrypkey().
Maintenance	2	02	Set this to 0 if you simply want to check the state of the authorization. Set this to 1 if you want to let the user change his or her authorization using cui2.exe. Set this to 2 if the program is ending.

The information you are sending to and receiving from cui2.exe is the same information as described in the CrypKey function calls.

To transfer information:

1. Assemble the information required into the PassedOut structure, including two random numbers.
2. Write the information to a file named cui2_infile.tmp.
3. Run the program cui2.exe and wait for it to complete.
4. Once cui2.exe completes, read in the information from a file called cui2_outfile.tmp.
5. Perform the **CKChallenge()** calculation and compare this to the number that was passed out.
6. The information from this file can be used as if it came from the CrypKey function calls.

The test code and cui2.exe are automatically placed in the Crypkey.60\External\Cui2 directory when you install CrypKey. Testcui2.c/testcui2.exe is an example program written in C that uses external linking. Cui2.h is a C representation of the structures passed in and out of your program.

5.5.3 External Text Link

This is an optional method of linking CrypKey to your application and is useful if you have a DOS program that can't call C functions. This external link is designed for the many DOS programs that are more comfortable with writing and reading Comma Delimited ASCII text files. Such programs include DBASE, FOXPRO, and CLIPPER.

This system has the following files:

CUI2.EXE	- DOS "CrypKey User Interface" external license manager
CUI2.C	- source code for CUI2.EXE
CUI2.MAK	- MSC 7.0 make file for CUI2.EXE
TESTCUI2.EXE	- sample externally linked DOS program
TESTCUI2.C	- source code for TESTCUI2.EXE
TESTCUI2.MAK	- MSC 7.0 make file for TESTCUI2.EXE
CUI2.H	- header file with information for linking with CUI2.EXE

For complete information on what is passed to and from CUI2.EXE, see CUI2.H. To demonstrate and test CUI2.EXE, put it in the same directory with TESTCUI2.EXE, and then run TESTCUI2.EXE. Both executables are already located in the DEMOS directory.

5.6 Creating a Windows NT License Service Installation

To create a Windows NT License Service installation, include setupex.exe and cks.exe in your installation script. The file setupex.exe is an NT install program. The file cks.exe is a self-extracting zip file of the 6 required setup files.

Note: don't extract the cks.exe zip file—setupex.exe expects to see this file as-is.

If, for example, InstallShield is used, the script should ensure that both setupex.exe and cks.exe are copied into the installation directory (i.e., the directory where the user's program is installed). The InstallShield script only calls setupex.exe.

5.6.1 Error Codes

Setupex.exe returns an error code if it runs into a problem. This code is returned in one of two ways:

1. in the process termination code, which can be read by the process that launched setupex.exe, if this is supported
2. if (1) is not supported, the error code is written in ASCII to a file called setupex.xco, at termination

Some installation programs are able to receive this error code. Processing these codes is optional.

The reported errors that are likely to be caused by a bug in your installation are:

```
#define NTDRVR_INSTALL_ERR_NOT_NT           -1
#define NTDRVR_INSTALL_ERR_REMOTE_DRIVE    -2
#define NTDRVR_INSTALL_ERR_CANNOT_COPY_FILE -3
#define NTDRVR_INSTALL_ERR_CANNOT_RUN_CKSETUP -4
#define NTDRVR_INSTALL_ERR_CKS_EXE_MISSING -5
#define NTDRVR_INSTALL_ERR_CANNOT_RUN_CKS_EXE -6
#define NTDRVR_INSTALL_ERR_MISSING_FILE    -7
```

The following error is reported when you do not have permission to load driver files.

```
#define NTDRVR_INSTALL_ERR_REG_COULD_NOT_OPEN_ -8
SERVICEMANAGER
```

You can clear the above error by logging in as a local administrator of the machine.

The following errors are unlikely to occur and should be reported to CrypKey:

```
#define NTDRVR_INSTALL_ERR_REG_COULD_NOT_OPEN_SERVICE -9
#define NTDRVR_INSTALL_ERR_REG_COULD_NOT_REGISTER_ -10
DIRECTOR
#define NTDRVR_INSTALL_ERR_STARTSERVICE_COULD_NOT_OPEN_ -11
SERVICEMANAGER
#define NTDRVR_INSTALL_ERR_STARTSERVICE_COULD_NOT_OPEN_SERVICE -12
```

```

#define NTDRVR_INSTALL_ERR_STARTSERVICE_COULD_NOT_      -13
START_SERVICE

#define NTDRVR_INSTALL_ERR_STOPSERVICE_COULD_NOT_OPEN_  -14
SERVICEMANAGER

#define NTDRVR_INSTALL_ERR_STOPSERVICE_COULD_NOT_      -15
OPEN_SERVICE

#define NTDRVR_INSTALL_ERR_STOPSERVICE_COULD_NOT_      -16
STOP_SERVICE

#define NTDRVR_INSTALL_ERR_REBOOT_NEEDED                 -17

```

5.6.2 Run Mode

CrypKey SDK has two run modes—silent and verbose.

In silent mode(`SetupEx.EXE /S`), `setupex.exe` does not put messages on the screen. It does complete checking and installs the Windows NT driver if it can. If it cannot install the driver, it returns an error code. Since silent mode does not report errors, it is up to the installation program to handle them. Note that the `REBOOT_NEEDED` message is not an error, but rather a warning.

In verbose mode (`SetupEx.EXE`), `setupex.exe` displays any errors it encounters. It does complete checking and installs the Windows NT driver if it can. If it cannot install the driver, it returns an error code. Note that verbose mode does not report errors onscreen if it encounters Windows 95/98/ME, but instead does nothing and silently returns.

5.6.3 Installation Strategies

Simplest Method

For any Win32 platform, the easiest installation strategy is to copy the files into the installation directory and run `setupex.exe`. The driver sets up the service, if it should be set up (i.e., it is a local Windows NT directory). If there is a problem, the driver reports it to the user, thereby doing all the necessary interface work for you.

More Complete Method

To provide a more complete package, you may wish to add some logic for someone trying to install onto a server from a client. Although your program may be installed this way, the driver cannot and users must be warned of this. You can avoid this issue by disallowing client to server installations or by offering a way to run only the driver install on the server.

Applying More Control

If you need more control over your installation, you can use our Windows NT installation silent mode by calling `SetupEx.EXE /S`. You then must retrieve the error codes and handle all errors yourself.

5.6.4 Uninstalling the Driver

The Windows NT driver has the ability to do partial and complete uninstalls. If other directories are using the driver, the uninstall removes only the directory `setupex.exe` from the configuration. If no other directories are using the driver, the uninstall stops the service and remove all the files used by the driver.

Procedure

1. Locate `setupex.exe` in the directory you want to uninstall.
2. From the directory, run `SetupEx.EXE /D`.

This performs all required actions. If no other directories are using the driver, all driver files are deleted and a message notifies the user that the computer must be rebooted before another installation can be done. You can avoid this message by using the silent mode (`SetupEx.EXE /D /S`).

Note that if the user attempts another installation before rebooting and the service was removed, the following “Error -9” message appears:

```
Register Directory: Unable to open CrypKey License Service.  
Reason: The specified service does not exist as an installed  
service.
```

The above error message disappears after the computer is rebooted.

If you wish to uninstall even if there are other directories registered, you can use the `SetupEx.EXE /U` command.

WARNING: this command leaves the other registered directories without a driver to serve them. It is recommended that the command option `/D` be used in release versions, and that option `/U` be used for test purposes only.

The driver can be reinstalled after the `/U` option is run, but you must first reboot. The other registered directories, if any are present, are still configured.

5.6.5 Testing

For a simple test, place both `SetupEx.EXE` and `CKS.EXE` in any directory, and then call `SetupEx.EXE`. You now see your directory in `Crypkey.ini`. If the `CrypKey NT` service had never been on your system, you would see that too.

5.6.6 Supported Versions of Windows NT

`CrypKey` supports Windows NT 3.5 (Build 807), Windows NT 3.51 (Build 1057), and Windows NT 4.0 (Build 1381). Only the Intel binaries are shipped. Contact `CrypKey` for up-to-date platform support. Windows NT versions predating version 3.5 are not supported.

Protecting Your Software with StealthPLUS™

An important complement to encrypted licensing is the methodology, embodied in StealthPLUS™, to protect your installations against hackers.

S

tealthPLUS™ is an important component of software security and the CrypKey SDK. You may implement SDK functions in your software, but without anti-hacking protection, an average skilled hacker can quickly change your code to skip the security.

Version 6.0 StealthPLUS™ has evolved to the highest level of security yet. Although no one can guarantee that your program can't be hacked, we can guarantee that StealthPLUS™ makes your program extremely tedious and painful to hack. Any version of your program that is hacked will not likely operate properly.

StealthPLUS™ is a strong deterrent to reverse engineering and patching, helping to ensure that the security logic you put in your program stays in your program. In less than 60 seconds, StealthPLUS™ automatically implements a battery of security measures in your program.

StealthPLUS™ protects your program in two ways:

1. It compresses and encrypts the program. Your executable file is smaller after you have “stealthed” it. Since the code is actually encrypted, reverse engineering and debugging of the file becomes extremely difficult.
2. It guards against patching while your program is in memory. It is technically possible to debug or redirect program execution (patch) around the security code while it is in

memory. StealthPLUS™ detects debuggers and patches and immediately halts program execution.

StealthPLUS™ also completely rewrites your program and runs it in memory in small pieces, minimizing the amount of code that must be exposed at any given time. This process provides an extremely strong level of protection.

StealthPLUS™ is simple to use. Simply tell it the input and output file paths and it does the rest, rewriting your EXE or DLL and building the above security features into them.

For your convenience, we have included two versions of StealthPLUS™:

Note: To run either version of StealthPLUS™, the Site Key Generator must be authorized.

- User Interface StealthPLUS™ (Stelthui.exe)

This version has a windows user interface that allows you to browse and save the input and output file paths.

- Command Line StealthPLUS™ (Stelthcm.exe)

This version allows you to easily automate the StealthPLUS™ process by adding StealthPLUS™ to your make files. It accepts the input and output paths as commandline parameters. The Stealth command syntax is as follows:

```
Stelthcm.exe {input filepath} {output filepath}
```

where {input filepath} is the EXE or DLL you want to stealth and {output filepath} is the stealthed file. The file name can be the same if it is put in a different directory.

The command line version is designed to allow you to automate the stealth task by adding it to the end of your compiling process. Many compilers support a custom step like this, so consult your compiler's documentation for full details.

6.1 Visual C++ Custom Step Example:

Procedure

1. Create a subfolder named `Treated` under the folder that your compiler outputs the target program or DLL.
2. In the VC++ IDE, select the menu item **Project\Settings**, then scroll the tabs to the right until you see on the **Custom Build** tab.
3. Enter the following:

Description: Stealth

Commands: `c:\CrypKey.60\stealth\stelthcm.exe`
 `$(InDir)\your.dll`
 `$(OutDir)\treated\your.dll`

Note: in the above, the command `InDir` can be written as `InputDir`. The parameter `your.dll` can also be given as `your.exe`.

Outputs: `$(OutDir)\treated\your.dll`

Note: in the above, the parameter `your.dll` can also be given as `your.exe`.

Completion of the above sequence causes VC++ to put a “stealthed” copy of `your.dll` or `your.exe` in the `Treated` directory.

6.2 StealthPLUS™ File Auto-Distribution

StealthPLUS™ (both the interface and command-line versions) have a useful additional feature: they can pack all CrypKey distributable files, as well as your files, into your executable.

StealthPLUS™ can even install the CrypKey License Service upon first run for you. For instructions on how to use this feature of StealthPLUS™, see Sec. 5.6, Creating a Windows NT License Service Installation.

The Auto Distribution feature ensures that:

- your EXE has all of the files it needs to run, no matter where it is
- your EXE always uses the correct version of associated files
- the NT Driver is installed
- your EXE is self contained and easy to use

6.3 Using the Interface (STELTHUI)

Procedure

1. In the CrypKey.60\Stealth directory, click the `stealthui.exe` filename. The system displays the following window (with the Site Key field blank):

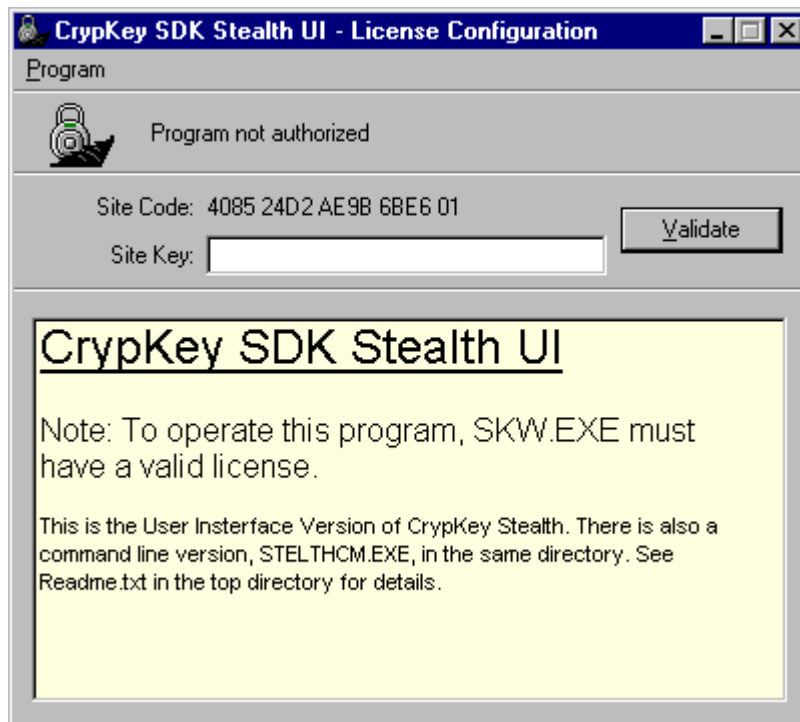


Figure 48: StealthPLUS™ Opening Window

2. In the Site Key field, type in your CrypKey Site Key, and click the **Validate** button. If the Site Key is valid, the system displays the following popup window:

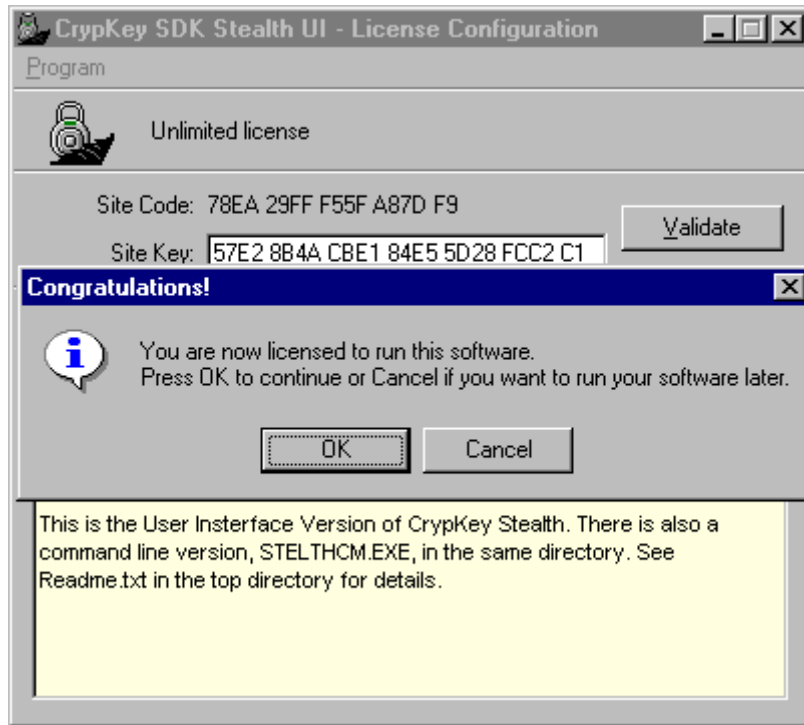


Figure 49: StealthPLUS™ Validated Window

In the above window, notice that the site code has now changed. This is standard in CrypKey. When CrypKey validates a program using a site key generated from the program's site code, it replaces the site code for that program.

3. In the above window, click **OK** if you wish to proceed. The system displays the following window:



Figure 50: StealthPLUS™ Configuration Window

4. In the above window, use the **Pick File** buttons to browse your directories and enter the following information:
 - in the **Program to compress** field: the full path of the source dll or executable file that you wish to protect.
 - in the **Output program** field: the full path of the protected dll or executable file that is to be distributed. If you want to place this file in the same directory as the source file, you must use a different file name.
5. Optionally, select the checkbox **Autoinstall NT drivers** if you wish to have StealthPLUS™ automatically install the CrypKey License Service drivers when your customer installs your product.
6. Optionally, select the checkbox **Attach and autoextract files in this folder**. This activates the **Pick folder** button, which you can use to find and enter the name of a

folder. During compression, StealthPLUS™ extracts files from this folder and pack them into the executable file for your application.

Note: any CrypKey files placed previously in the directory AutoDist are now packed. If there are files you do not need, simply remove them before running the StealthPLUS™ interface. Similarly, other files that you have placed in the UserDist directory are now packed.

Note that StelthUI.Exe allows you to save the configuration and specify a different user directory for each EXE.

7. Use the **Pick folder** button to select the folder where the NT drivers and auto-extracted files are placed.
8. To save your StealthPLUS™ configuration, click the **Save** button. If you are not planning to stealth your software during the present session, you can close this window.
9. When you are ready to stealth your software, click the **Begin compression** button.

Note: if you have started a new StealthPLUS™ session, having previously performed Steps 1 to 6, the StealthPLUS™ Configuration window appears as above (Figure 50), ready for you to click the **Begin compression** button.

The system now displays the following popup window, overlaying the StealthPLUS™ Configuration window:

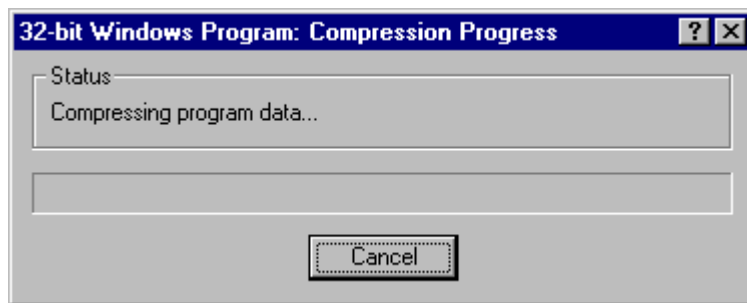


Figure 51: StealthPLUS™ Compression Begun

6.4 Using the Command Line Version (STELTHCM)

Procedure

Syntax

The syntax for the command-line version of stealth is as follows:

```
stelthcm.exe [-n] [-a directory] infile outfile
```

where

-n is the option for Autoattach NT drivers

-a is the option for Attach and autoextract files in a given directory

infile and *outfile* must be given as different filenames

If a path has spaces in it, surround it with double quotes—“ ”.

Results

When your program runs, the temporary CrypKey files, such as the NT Driver, are unpacked to a temporary directory, run, and deleted. All other files are unpacked to the EXE's directory before the EXE is run.

Sample “Stealthed” Program

A sample “stealthed” program, together with packed CrypKey and user files, is in the CrypKey.###/Demos/SampleStealthed directory.

Note that running this program unpacks all files.

The configuration used to create this program is saved as the first configuration in StelthUI.Exe.

Note: This configuration is correct only if CrypKey is installed to C:\CrypKey.###.

Counter-Hacker Strategies

The measures presented here are not guaranteed unbreakable, but are designed to deter license system tampering and reveal tampering when it occurs.

There are some measures you can take when developing your software to help prevent users from tampering with your product. Ensure your software responds with error codes when a function fails.

Note: these measures are designed for environments that do not include CrypKey StealthPLUS™. When distributing your software, use the StealthPLUS™ program that is included in CrypKey SDK. For details on using StealthPLUS™, see Chapter 6.

7.1 Anti-tampering Measures

The anti-tampering measures include features that:

- prevent the license system from being compromised through the substitution of an impostor license system (i.e., one that always grants all license requests)
- prevent the license system from being compromised through multiple patches (there is no set of patches that can defeat the license system for all applications)
- prevent, through a set of guidelines, any single application's license verification subsystem from being defeated by any single patch
- require an obvious and intentional effort to defeat for each application

The following guidelines may go against some accepted coding practices. It is your choice whether or not to implement them.

- Do not compare the authorization result immediately after calling a **CrypKey** function. Instead, save the result and compare it later. This is security by obscurity.
- Do an extra check on authorization in some obscure but frequently used part of the program. If the result is unauthorized, obtain the system time, do nothing until the time is an even multiple of 5 (for example), and then disable something important.
- Call the **CKChallenge()** function, especially when using a **CrypKey** DLL or external linking. This is a very difficult function to hack.
- Incorporate an internal and obscure checksum over the code that deals with the license system and with the verification. Obscure the checksum code if you can.
- If applicable, take over any debug and single-step vectors that might be used by debuggers. Take great care when using such an advanced measure.
- Add random items to the application, such as meaningless reads, compares, and subtractions. In situations where a hardware monitor is used, this may increase the number of hardware breakpoints that occur.
- Verify the result more than once. Provide a faux compare immediately after the call to **GetAuthorization()**.
- Don't do a comparison with a single compare. Instead, perform some mathematical operations on the result, computing another result that is verified later in the code.
- For platforms that permit self-modifying code and where the actual challenge algorithms are included in the code, hide the challenge algorithms by encrypting the code. Decrypt only long enough to verify a challenge and then re-encrypt it. Checksum the code that calls the encrypt/decrypt routines to prevent alteration. Have the entry point in the license module verify the code offset (if applicable) from which it was called. Encrypting the algorithms makes them more difficult to locate in the code. Also, encrypt the Master Key and the User Key.
- If the host operating system permits, place most of the authorization code and data into overlays that can be discarded.

- Call **GetAuthorization()** periodically (i.e., every 15 minutes).

These are just a few possibilities to consider. Think about anti-virus techniques and the measures taken to prevent code modification. Those same anti-tampering techniques are applicable here.

7.1.1 Error Codes

Always have your program respond with specific error codes when a function fails. SLAPI has numerous functions with many different error codes. When something goes wrong with a particular function, it is a tremendous help to know specifically which function and operation is failing. The problem can be determined and resolved much quicker when this information is available.

Always call **InitCrypkey()** and **GetAuthorization()** before any other CrypKey functions.

Always check the return on **InitCrypkey()** and do not let the program run if the return is less than zero.

Ensure **EndCrypkey()** is called no matter how your program is exited. It is a common mistake to call only **EndCrypkey()** in your File/Exit menu routine. If the user exits by double-clicking the title bar, the **EndCrypkey()** function is skipped, which causes problems.

Network Licensing

CrypKey allows you to control the licenses you issue, even when your software is distributed across a network. If more clients request access to your software than you have allowed, CrypKey provides access on a first-come, first-served basis.

Y

ou can control the number of users and computers that use your software concurrently from a networked server. You can transfer your licenses from machine to machine across a network.

If there are more requests for access to your software than you have allowed, CrypKey queues the requests. For example, if you issue a key with the Site Key Generator that allows four floating licenses, your customer can run your program on any four clients in his or her network. If a fifth client attempts to start your program, he or she is told to wait until someone else currently running the program closes it. If more users attempt to start your program, they are placed in a first-come, first-served queue. The clients each get their turn in the order their access attempt was made.

8.1 Implementing Floating Licenses

It is possible to use CrypKey on any MS-DOS, Windows, or Windows NT-based computer on a network. The network operations have been made transparent as much as possible. There are three things you must do to change your single-user application into a multi-user application: provide your program with multi-user functionality, install the appropriate network driver, and specify a floating license type.

8.1.1 Providing Multi-User Functionality

If you intend your program to be used by more than one user concurrently over a network and you intend to limit the number of users who can use your program concurrently, you must call **GetAuthorization()** at regular intervals. The call to **GetAuthorization()** advises CrypKey as to whether the user is still using the program. If CrypKey suspects that a user is not using the program (perhaps the machine has crashed or the network connection has failed) it takes the license, after the period of the `network_max_checktime` is exceeded, and gives it to a user who has requested access. This ensures that the licenses are not permanently held when the users are not using the program.

The recommended interval between calls to **GetAuthorization()** is 5 to 30 minutes, depending on your application. The `network_max_checktime` parameter, located in the **InitCrypkey()** function, is the maximum time (in seconds) that CrypKey should wait before releasing a license that is no longer accessed. `Network_max_checktime` should be two or three times the expected interval between calls.

The **FloatingLicenseSnapshot()** function provides a snapshot of all users holding or waiting for a network floating license.

8.1.2 Installing a Network License Server

The server is the computer that has your program on its hard drive. For network licensing, CrypKey-protected programs require a driver to run on the computer that acts as a server.

Normally, users read your program from a client computer that is connected to the server via a network. You must install the appropriate network driver on the network server that contains the license file for your protected program. Once the driver is running, CrypKey operates on the server from any other computer, just as if the server were a local drive. For more information on installing network drivers, see the installation instructions in Sections 2.2 to 2.5.

8.1.3 Specifying a Floating License Type

The Site Key Generator enables you to specify a floating license, intended for use over networks. You must provide a floating license if the program is to be used over a network. A fixed, or node-locked, license cannot be used over a network; an error is generated if this is attempted.

8.2 CrypKey Network Drivers

The /NETDRIVE directory contains the NT CrypKey Driver and Network drivers for other server platforms.

Note: The NT driver is required for *standalone and/or network licensing* on all NT platforms, including XP, Win 2000, and NT. All other drivers in the /NETDRIVE directory are required only for network licensing.

CrypKey uses an encrypted file request/response model to handle network authorizations. This innovative approach has major advantages:

- Your application uses the same CrypKey library for all network platforms. No changes to your application are required for different networks.
- Since your application uses only one library for all networks, there is a significant savings to you in code size and complexity.
- Your program works on any network that does file transfers.

8.2.1 Supported Platforms

Five network server operating systems are supported and each requires a different driver. These drivers are listed below, in the order of likelihood that you will use them:

- `setupex.exe` and `cks.exe`: These driver files are required to run an application on Windows NT (including Windows NT/2000/XP) standalone or server machines.

Note: you must always use this driver for both standalone and network installation when running NT platforms.

- `ckserver.nlm`: This is a Novell NetWare Loadable Module that works for NetWare 3.x, 4.x, 5.x, and 6.x. Ckserver.nlm does not require any configuration. Novell supports only network use.
- `wckserve.exe`: This is a Windows program that runs on any Windows 3.x, Windows 95, Windows 98, or Windows ME server. It is unlikely that you need to support this platform as a server.
- `ckserver.exe`: This driver is intended for an MS-DOS server. It is a terminate and stay resident (TSR) program and cannot be run on a Windows server. It is highly unlikely that you need to support this platform as a server.

- OS2: This driver is included for legacy support.

All server drivers require the client to have read and write access to the shared directory. For NetWare servers, the client must have read, write, filescan, modify, and delete permissions.

8.2.2 Installation Procedures

NT CrypKey Driver

This driver is required for standalone or network licensing use on NT Platforms.

***Note:* This is the most important driver of the CrypKey system. It is the Crypkey License Service required on NT/2000/XP systems.**

Procedure

1. Copy the files setupex.exe and cks.exe to the application directory.

***Note:* You require administrator privileges to install the license service.**

2. Run setupex.exe.

setupex.exe extracts and installs the following files from cks.exe:

- WINNT/System 32 directory: ckldr.sys, crypserv.exe
- WINNT directory: ckrfresh.exe, setup_ck.dll, setup_ck.exe

Note: In Windows XP, these files go into the Windows and System directories.

To measure the success of the installation, a file is created in the application directory called setupex.xco. It contains only the value 0 if the installation is successful.

See the readme.txt in ~\CRYPKEY.60\NETDRIVE\NTDRIVER for details that have been documented after publication of this manual.

Novell Network Driver – CKSERVER.NLM

Procedure

1. Copy CKSERVER.NLM to the System directory.
2. In the Novell AUTOEXEC file, insert the following statement:

```
LOAD CKSERVER
```

This starts the driver upon reboot.

3. To start the driver manually, on the Novell server in monitor mode, type

```
LOAD CKSERVER
```

Note: Novell's Patch Level #5 for Novell 3.x. is the minimum patch level required to run CKSERVER.NLM on 3.x. This is available from us by request.

Windows 3.x or WIN95 – WCKSERVE.EXE

This is a Windows program that runs invisibly in the background. You must provide it the path where your CrypKey-protected application resides. Directories must be defined in the command line or by setting the environment variable CKSERVE. (This is not a misprint — the environment variable is CKSERVE.)

Example 1 – command line

```
WCKSERVE.EXE C:\APP1;C:\UTIL\APP2;E:\WIN\APP3;
```

Example 1 – autoexec.bat

```
SET CKSERVE=C:\APP1;C:\UTIL\APP2;E:\WIN\APP3;
```

To install this on your customer's machine, ensure it is restarted each time the machine is rebooted by referencing this in the Windows StartUp folder.

DOS – CKSERVE.EXE

This is a DOS TSR that runs invisibly in the background. You must provide it with the path to the directory where your CrypKey-protected application resides. Directories must be defined on the command line or by setting the environment variable CKSERVE.

Example 1 – command line

```
CKSERVE.EXE C:\APP1;C:\UTIL\APP2;E:\WIN\APP3;
```

Example 1 – autoexec.bat

```
SET CKSERVE=C:\APP1;C:\UTIL\APP2;E:\WIN\APP3;
```

To install this on your customer's machine, ensure it is restarted each time the machine is rebooted by referencing this in the AUTOEXEC.BAT.

OS2 Driver

Windows CrypKey protected programs can be run on an OS2 machine and networked from an OS2 machine using the OS2 driver.

- CKSVROS2.EXE - OS2 driver native OS2 program
- CKSVROS2.HLP - OS2 driver help file (OS2, not Windows, help file)
- CKOS2WIN.HLP - OS2 driver Windows help file

Moving Protected Programs

The license transfer system is an optional feature that allows your clients to transfer their licenses from one computer or directory to another without contacting you.

If multiple-copy fixed licenses are issued, we recommend using the license transfer feature to allow your customer to distribute the multiple copies to different sites.

There are two methods for moving a CrypKey license:

- **Direct Transfer:** Used to move a license from one local or networked directory to another.
- **Floppy Transfer:** Used to move a license from one computer to another when the two computers are not networked or the CrypKey network driver is not present. In this case, the move is done using a floppy disk.

These two methods do not actually copy the program from one directory to another. Instead, they move the license from one program to a copy of the same program in a different directory. The user must therefore copy the program to the target directory or computer before using the license transfer function.

9.1 Direct License Transfers

The **DirectTransfer()** function must be run from a program that has a valid license (the source program) and given a directory path to an existing copy of the same program that does not have a valid license (the target program)

The function simply transfers one license from the source directory to the target directory. The target directory must be local (i.e., on a drive located in the same computer) to the source directory or the directories must be connected by a network and have a CrypKey network driver servicing each of them.

9.1.1 Example

If you have a program with one valid fixed license in C:\Temp and you want to move it to C:\Apps\Bingo:

1. Copy your program to C:\Apps\Bingo.
2. Run the program that is in C:\Temp. Call **DirectTransfer()** and provide it the path name of "C:\Apps\Bingo". Instantly, the copy in C:\Apps\Bingo has a valid license and the copy in C:\Temp does not.

9.2 Floppy Disk License Transfers

The floppy disk license transfer process involves transferring the special license files and is handled entirely by the CrypKey system. At no time during the transfer process is your license vulnerable to reproduction by bit copiers, file duplication, or file tampering.

To transfer a license from one computer to another, you must have a copy of the program installed and authorized on one computer and a copy of the program installed but not authorized on another. Assuming that the transfer is from an already-licensed source computer to an unlicensed target computer via floppy disk:

1. Start the program in the target computer and register the transfer by telling the program to call the **RegisterTransfer()** function. The user must supply the path to the floppy disk. This function puts a registration imprint file on the floppy disk. Take out the floppy disk from the target computer and place it in the source computer.
2. Start the program on the source computer and tell it to call the **TransferOut()** function. The user must supply the path to the floppy disk. This function reads the registration imprint file and writes a matching license imprint file to the floppy. It also discontinues the license at the source if it had one copy or it decrements the license at the source by one if it had multiple copies. Take out the floppy disk from the source computer and return it to the target computer.

3. Restart the program in the target computer and tell the program to call the **TransferIn()** function. The user must supply the path to the floppy disk. This completes the transfer and discards the intermediate imprint files.

If you want your program to be able to transfer its license via floppy disk, you must program your application to allow the user to initiate the following functions:

- **RegisterTransfer()**
- **TransferOut()**
- **TransferIn()**

Frequently Asked Questions

The following questions are frequently asked of our technical support representatives.

S

uccessfully answering your questions is a priority of the CrypKey SDK technical support team. Since CrypKey's inception in 1992, we've striven to eliminate your guesswork. We encourage you to read through the following pages before contacting us for support.

10.1 General

10.1.1 Software Protection



Q Is illegal copying of my product a real concern ?

A Yes. In today's high-tech world, you need copy protection. A copyright sign on your product does not guarantee that users will not copy your product. Copyright laws, like all others, can be broken despite enforcement efforts. If you're considering marketing overseas, the situation is even more serious. Many countries outside North America are without any form of copyright law, further increasing the possibility for unpaid copies of your product to enter the market.

Although software giants such as Microsoft can absorb sale losses related to illegal copying, smaller players, whose market is more defined and vertical, cannot. Software is no longer solely produced by large software houses and sold in mass quantities. Today's software developers include doctors, lawyers, engineers, and those in your profession: essentially any individual with knowledge that can be shared and marketed through the medium of computers.

10.1.2 CrypKey Advantages

- Q Why is CrypKey the protection package of choice?
- A The CrypKey licensing system circumvents the problems prevalent in current software protection technology, as outlined below:

Table 9: Software Protection Technology – Problems and Solutions

Copy Protection Problem	Description	CrypKey Solution
Key failure (hardware (dongle) or disk keys are subject to higher rate of failure than computers)	Keys transfer software from one computer to another. Hardware keys can suffer from fatal static shock and diskette keys can suffer from media failure or be misplaced. Because users frequently transfer software, this action, while protected, must remain as convenient as possible.	CrypKey replaces hardware and disk keys with telephone or email authorization, where client computers become the key so that clients can easily transfer your protected product to other computers. With CrypKey, your master disk is MS-DOS copyable and is not required for product transfers.
Hardware incompatibilities	Hardware keys can be incompatible with some computers, printers, or other hardware keys. Disk keys can be incompatible with certain makes of computers and different sizes of disk drives.	No hardware key means no hardware incompatibilities. CrypKey has an outstanding record of running on any PC-compatible computer, including foreign machines (i.e., Japanese NEC PC) and diskless workstations.

Copy Protection Problem	Description	CrypKey Solution
Induced downtime	When a copy protection key fails, time is required to obtain a new one. This depends on key inventory, protection company replacement procedures, and client location.	Your clients are up and running after one quick telephone call rather than waiting for a hardware or disk key in the mail.
Failure to protect	Bit copiers, sold as programs that can make backup copies of copy-protected software, can easily copy most disk keys.	With CrypKey, your product cannot be copied by a bit copier. Your product is shipped on an MS-DOS copyable disk and runs in demonstration mode until telephone authorization, whereby it becomes fully functional. Because your disk does not need to be copy protected, you can manufacture your disks yourself.
Costs	Key problems can eat up more than 50 percent of your technical support allowance.	Non-technical staff can issue telephone authorization with the CrypKey Site Key Generator, freeing up your technical staff for technical support.
Client annoyance	Market acceptance of hardware keys (dongles) is declining.	Deliver product upgrades and renew licenses instantly. CrypKey allows you to specify levels and/or options and the duration or number of times clients can use your product. You also have the option of selling your product on a time or per-use license basis.

10.1.3 Telephone Authorization

Q How does CrypKey protect products without using keys?

A Telephone authorization replaces keys in the CrypKey system. The process is as follows:

- once installed on your client's computer, your product generates a site code specific to that computer, which your client sends you by telephone, email, or fax
- using CrypKey, you generate an authorization code (for immediate or later distribution to your client) that allows your product to run only on the computer that generated the site code

10.1.4 Client Annoyance

Q Won't customers be annoyed by needing to make a telephone call to use my product?

A CrypKey users have advised us that this drawback is typically perceived rather than actual, mitigated by client relief over the absence of hardware or disk keys. However, ways to capitalize on the CrypKey telephone, email, or fax authorization system include:

- Ship your product to potential clients in demonstration form. When your clients are ready to buy, use the authorization interaction to answer questions and get customer feedback.
- Send your clients a disk that demonstrates your product's options. Ask clients to call when they have determined which options they want. With CrypKey, you can turn on the desired options during the call, and clients often purchase more options than originally ordered.
- Ensure your client is comfortable and satisfied with running your product. Clients appreciate support and you may prevent problems and ill will from occurring.

10.1.5 24-hour Telephone Support

Q Do I need 24-hour telephone support?

A No. CrypKey's trial licensing feature ensures that clients can contact you for permanent authorization during regular business hours and can open and use your product any time they choose, including weekends and holidays. CrypKey allows you to expedite the repair or modification of licenses. CrypKey dramatically reduces your customer support hours.

10.1.6 Computer Signatures

Q Is a computer's signature derived from the volume serial number? If so, hackers could alter this number with programs such as Norton Utilities or PC Tools or those that directly access hard disk sectors. If not, how difficult is cloning the machine authorized to run my product?

A Computer signatures are derived from unchangeable parameters particular to the machine's software installation, which are nearly impossible to duplicate on another machine.

10.1.7 License Transfer to Floppy Disk

Q Transferring the license for my CrypKey-protected product to disk seems dangerous. A pirate could disk copy the floppy to generate an exact image of the first that could easily be used to authorize other machines. Is the onus on me to enable or disable the transfer to floppy?

A Although the decision to allow floppy disk license transfers is yours alone, note that a transfer incorporates the following process:

1. The signature of the target computer is written on the floppy (in highly-encrypted files).
2. The license is transferred from the source computer to the floppy.
3. The license is transferred from the floppy to the target computer.

Even if the floppy is copied after Step 2, it only works in the specific target computer because of information encoded in Step 1.

Q What if the transfer to the floppy is corrupted? Does that mean the license is deleted from the source machine, but is unreadable from the floppy?

A Yes. We recommend that customers minimize the number of times the transfer is done, and take precautions with their media. There is little chance of this happening with Direct Transfer, since the process includes many safety tests.

10.2 Technical

10.2.1 Disk Compression Programs



Q Is CrypKey affected by disk compression and defragmentation programs such as Norton Utilities Speed Disk or Nuts and Bolts?

A CrypKey is generally immune to these programs. With Speed Disk, however, adhere to instructions in *Norton Utilities Speed Disk*, page 62, to ensure CrypKey licensing files remain undamaged (see Section 11.5: Norton Utilities Speed Disk for more information).

10.2.2 Watcom Support

Q Does CrypKey support Watcom C++9.5 and higher?

A Yes. CrypKey supports the Watcom development environment via our 16- and 32-bit DLLs along with MS-DOS programs with external linking capabilities.

10.2.3 Library Impacts

Q Does CrypKey affect other third-party libraries?

A No. CrypKey is highly non-intrusive, meaning that it works with virtually any software, and more importantly, any hardware.

10.2.4 Hard Disk Failure



Q What happens if the hard disk fails, thus invalidating the CrypKey license?

A We provide a new license, no questions asked. Our clients always receive the benefit of the doubt.

In reality, hard disk failures among CrypKey clients are rare. CrypKey's three-year relationship with Flowel, a popular flow calculation program we market worldwide, has resulted in eight reported hard disk crashes, of which we feel only two are suspect.

10.2.5 Product Update Releases

Q How does CrypKey manage product update releases?

A CrypKey simplifies releasing your product updates. Copy the new version of the product over the old one and it assumes the existing license.

10.2.6 Multiple Program Names

Q Can a single product have multiple modules/names in relation to one CrypKey license?

A Yes. To successfully award a product multiple names, ensure all names key off one file guaranteed to occupy the same directory as your product. CrypKey allows you to select the file you want the license based on.

For example, all versions of Product1 key off a file named product1.dat, located in the same directory as the .exe for every installation of your product. This method makes it easy to consistently install CrypKey in all your product versions.

10.2.7 Floating Licenses



Q My clients sometimes ran three floating copies of my simple Windows product when they were only licensed for two. How did this happen?

A CrypKey's mechanism for counting license numbers is foolproof. If a particular application of your product fails to call **Get Authorization ()** regularly, it loses its license. Parameters of the **InitCrypkey ()** call control the time before the application fails. Change these parameters to suit your requirements to prevent unauthorized use of your product.

10.2.8 Computer Clock Manipulation

- Q** Can clients tamper with CrypKey's time-dependent licenses by setting back their PC clocks?
- A** No. CrypKey records each time clients check their license status from within your product. If clients set back their clocks more than 75 minutes, CrypKey generates an error code for your product.

Additional Clock Manipulation Checks



There are programs available on the market designed to try to deceive time-restricted software protection systems. These programs automate the process of changing a computer's clock in order to bypass time-restricted protection. These programs change the time registered by the protected program. For example, if the registered time never extends beyond your program's trial period, a user can in theory run your program indefinitely.

CrypKey safeguards against clock manipulations by monitoring your customers' computers for the inconsistencies generated by such an attack. CrypKey detects when time is falsely reported to your program, and prevents your program from running. Times reported by CrypKey are that of the server, adjusted to the local time zone of the computer.

CrypKey may also detect clock manipulation if users adjust their computer clock by more than several hours, even for benign purposes such as testing. If your program is prevented from running in these situations, your customer can correct the situation by rebooting his or her computer.

10.2.9 CD-ROM Distribution

- Q** Does CrypKey support distribution via CD-ROM?
- A** Yes. CrypKey can be used if you install the executable to your hard drive from a CD-ROM or it remains resident on the CD-ROM.

In the latter case, the problem that arises concerns the inability of CrypKey to write its license files, which are created dynamically, to the read-only CD-ROM. Circumvent this issue by storing CrypKey license files on your hard drive. CrypKey stores license files where you specify.

In a network license situation, rather than downloading a large program across the network when it runs, install the program on the target machine and ensure only the license resides on the server.

10.2.10 Internet Distribution

- Q** Can files protected by CrypKey be distributed over the Internet?
- A** Yes. Because CrypKey eliminates hardware or disk key technology, it provides an ideal channel for internet distribution.

10.2.11 File Set Protection

- Q** Can CrypKey protect a set of files?
- A** Yes. Use CrypKey to protect an unlimited number of programs related to your product. The options feature individually enables or disables up to 32 programs per product at no additional cost.

You can set up sub modules to check the main modules license file. You achieve this by passing the main module path to the **InitCrypkey()** call in **all** modules. All modules must also use the main module's Master and User Keys.

10.2.12 Foreign System Compatibility



- Q** Is CrypKey compatible with foreign PCs and operating systems?
- A** Yes. CrypKey is fully functional on the Japanese NEC PC and the Kanji version of Windows 95. Other Asian and European versions of Windows are also supported.

10.2.13 Removable Hard Drives

- Q** Does CrypKey work with removable hard drives (i.e., Iomega Zip and Jaz drives)?
- A** Yes. CrypKey fully supports Iomega Zip and Jaz drives. CrypKey-protected products can be normally installed and run from these drives. Other removable hard drives, however, are not supported and CrypKey-protected software does not run from them.

10.2.14 Anti-Virus Software

Q Will CrypKey trigger anti-virus software that is in use on a computer?

A No. CrypKey does not conflict with any anti-virus software.

10.2.15 Installer Software

Q Is CrypKey compatible with such installer software as InstallShield3?

A Yes. CrypKey is fully compatible with all installer systems, especially InstallShield3. We use InstallShield3 in our own software and have never experienced problems.

10.2.16 US Exports

Q Can I export my CrypKey-protected product from the United States?

A Yes. CrypKey complies with U.S Department of Commerce rules for export to all countries except Cuba and Syria.

10.2.17 Software Certification

Q If I integrate CrypKey into my software, will I have difficulties getting my software certified?

A No. CrypKey supports all elements required to allow your product to qualify for Microsoft's Designed for Windows certification.

10.2.18 Windows 98

Q Does CrypKey work on Windows 98?

A Yes. CrypKey and CrypKey-protected products run under Windows 98. Note, however, that if you are converting from FAT 16 to FAT 32, you must first transfer your license to another computer to retain your authorization.

Troubleshooting

Having trouble? This section provides solutions to some common problems.

Since implementing CrypKey SDK can create problems in some environments, you should check your environment type against the listed types to pinpoint problems and identify solutions.

This chapter includes the most common problems reported to CrypKey's technical support team. For problems not listed here, please visit our website at www.crypkey.com and review our searchable support database. The database is accessible using a customer password that you can obtain from us.

11.1 CrypKey Error Messages

Most CrypKey operational problems are manifested by error messages. These errors can be general (not specific to functions) or **specific to functions**.

11.1.1 General Errors

CrypKey general errors are listed in Sec. 12.2: General Error Values.

General error codes are in the ranges –100 to –199, and –200 to –299. General errors include network and DLL communication errors. You should always check for these return values and give some explanation to the customer if they occur. **ExplainErr()** (Sec. 12.3.8) returns an appropriate message.



Note: errors in the –200 to –299 range result from security checks. The most common cause of these errors is the placement of the .ngn file in a different directory than the one the executable resides in. In these cases, moving the .ngn or executable to the proper location should clear the error. If it does not, please contact CrypKey technical support.

The most common CrypKey error is Error 102 with the message NETWORK_NO_CKSERVE. See Sec. 11.2 for a detailed discussion of this error.

11.1.2 Function Errors

You'll find details of function-specific error codes in Sec. 12.3: Function Descriptions. See also Sec. 14.2: CrypKey Return Codes for a quick reference of errors generated by the **InitCrypkey()** and **GetAuthorization()** functions. These two functions are required in all CrypKey operations and consequently errors generated from them tend to be more frequent and critical than errors generated from other functions.

11.2 General Error 102

As mentioned earlier, Error 102 (“NETWORK_NO_CKSERVE”) is the most common CrypKey error message. It tells you that CKSERVE.EXE (or its equivalent) is not serving the program. This means that the CrypKey driver you are using is not serving the directory where your CrypKey-protected application is located. All CrypKey-protected applications must be served by the driver. This error usually occurs on Windows NT.

If you encounter this error, send an email message containing the date of setupex.exe to CrypKey technical support at support@crypkey.com.

11.2.1 CrypKey License Service Problems

When there is a problem with the CrypKey License Service, the user reports one of the following:

- An error window such as the one shown:

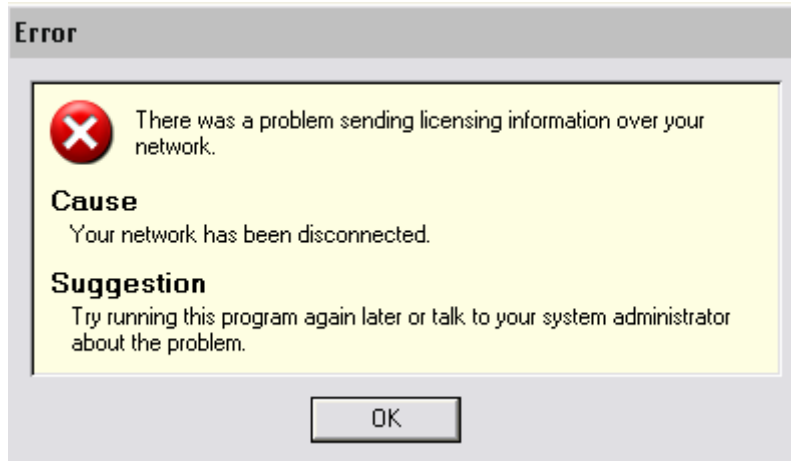


Figure 52: Network Disconnect Message Window

- A return code and error message from SDK:

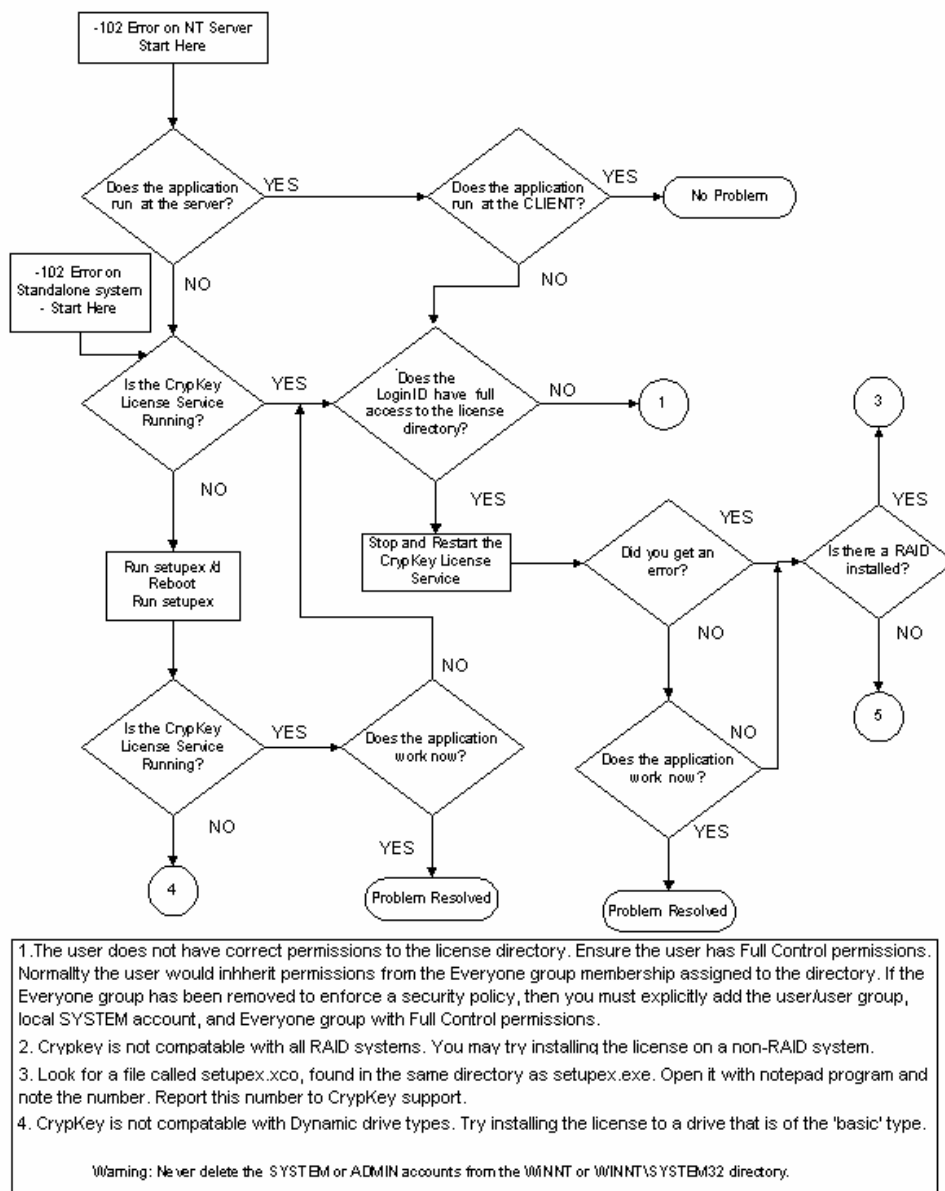
INITIALIZATION FAILURE

NETWORK: NETWORK DRIVER APPEARS NOT TO BE SERVING IN
DIRECTORY / -102

Problems of this type usually occur on NT or Novell Netware file systems where the directory security is modified. Refer to the following troubleshooting charts for assistance with this problem.

102 ERROR Trouble-shooting Flowchart

Applies to NT, Win 2000, and XP operating systems



support@crypkey.com

Figure 53: NT File System Troubleshooting Flowchart

102 error on Novell

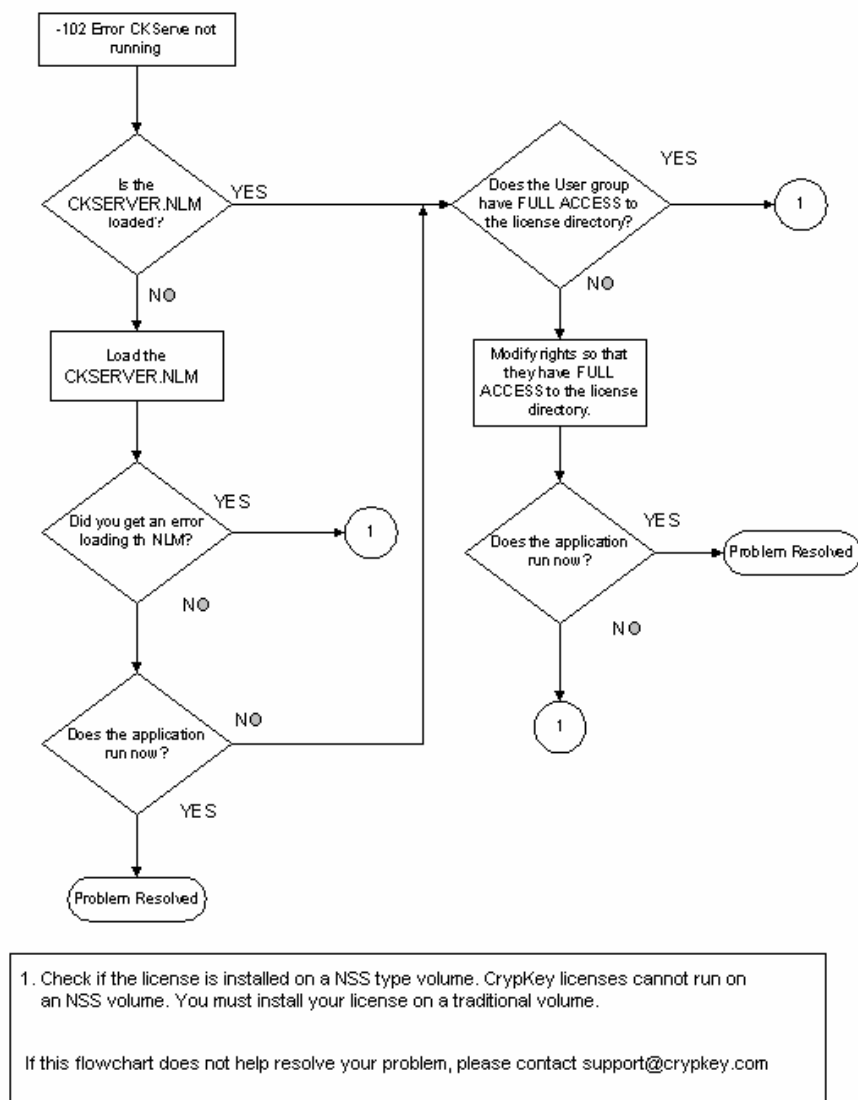
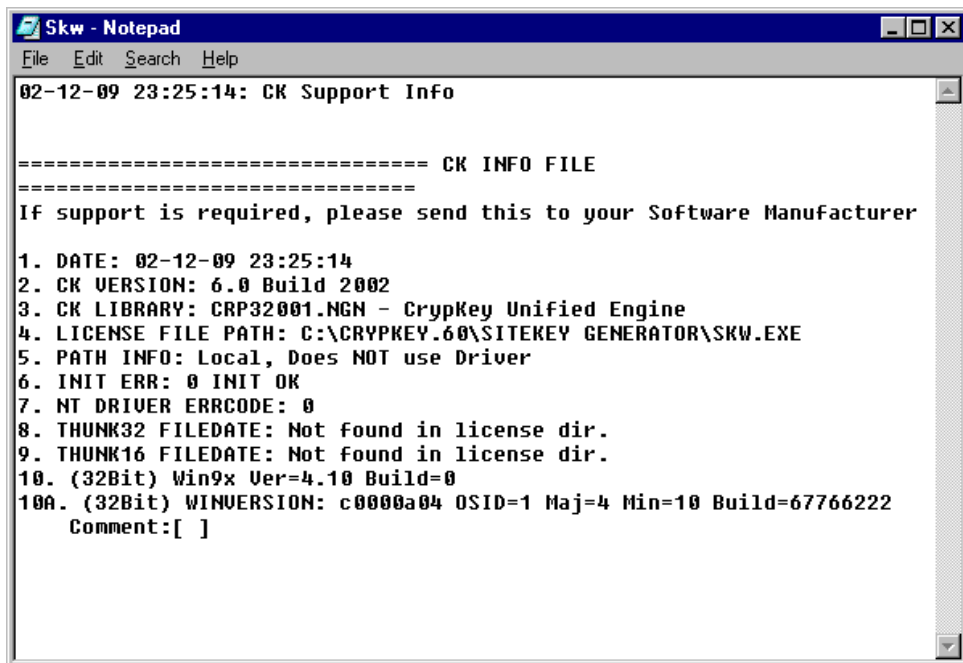


Figure 54: Novell File System Troubleshooting Flowchart

11.2.2 CrypKey Technical Support

When requesting technical support from support@crypkey.com, ensure to include the following information:

- company name
- CrypKey account number
- a brief and concise explanation of the problem
- error messages: your application's error messages may be ambiguous or not have significant meaning to CrypKey support. For this reason, develop your application to reference the associated CrypKey function and return code in your error messages.
- sequence of events required to recreate the problem. Include information such as:
 - order of CrypKey functions called
 - return codes from those functions
 - operating system
 - license type and duration
- .ckn file—this is a file created by the CrypKey functions and contains useful information that may be valuable for resolving your problem. The file resides in your CrypKey directories. When opened, the .ckn information appears in a window similar to the following:



```

Skw - Notepad
File Edit Search Help
02-12-09 23:25:14: CK Support Info

===== CK INFO FILE
=====
If support is required, please send this to your Software Manufacturer

1. DATE: 02-12-09 23:25:14
2. CK VERSION: 6.0 Build 2002
3. CK LIBRARY: CRP32001.NGN - CrypKey Unified Engine
4. LICENSE FILE PATH: C:\CRYPKEY.60\SITEKEY GENERATOR\SKW.EXE
5. PATH INFO: Local, Does NOT use Driver
6. INIT ERR: 0 INIT OK
7. NT DRIVER ERRCODE: 0
8. THUNK32 FILEDATE: Not found in license dir.
9. THUNK16 FILEDATE: Not found in license dir.
10. (32Bit) Win9x Ver=4.10 Build=0
10A. (32Bit) WINVERSION: c0000a04 OSID=1 Maj=4 Min=10 Build=67766222
    Comment:[ ]

```

Figure 55: .ckn Diagnostic File

- the CrypKey libraries used
- your compiler and version

11.3 Windows NT Diagnostics

The Windows NT driver maintains two error log files. You can help us rectify your Windows NT driver problems by sending (by email or fax) the error.log and the errorid.log files, located in the C:\WinNT and the C:\WinNT\System directories.

11.4 NetWare-related Issues

The following errors can occur on Novell networks using CrypKey ckserver.nlm.

11.4.1 Sample Error 1

If your clients attempt to execute your CrypKey-protected product from a NetWare server, an error message resembling the following occurs:

```
CKSERVE is not serving this directory
```

To solve this error, ensure:

- CKSERVER.NLM is running on the server
- the CKSERVER.NLM path statement within the autoexec.ncf is correct
- the directory name does not contain a period
- the maximum number of users for the CrypKey-protected product is not exceeded

11.4.2 Sample Error 2

If your referenced library routine cannot be located within the current .nlm, the loaded Novell clib.nlm library may not be current, generating an error message resembling the following:

```
Undefined symbol: _chk
```

To solve this error:

1. Implement the Novell patch libup5.exe or a newer patch (available from CrypKey or Novell).
2. Ensure a copy of directfs.nlm is loaded.
3. Shut down the server and restart. This loads the new clib.nlm (clib.nlm cannot be manually unloaded and loaded).

11.4.3 Sample Error 3

If none of your clients are logged onto the network executing the CrypKey-protected program, but an error stating that there are users logged on appears:

1. Delete the <programname>.tb2 file from the network.
2. Shut down the server and restart it.
3. Restart the application.

11.5 Norton Utilities Speed Disk

Speed Disk is the defragmentation utility included in Symantec's Norton Utilities. Unfortunately, Speed Disk moves CrypKey licensing files, causing license loss. CrypKey uses four special licensing files (.ent, .rst, .key, and .41s) to control your product licensing. These files are hidden system files and reside in the same directory as the protected product.

To prevent license loss:



1. Open Speed Disk, and choose **File, Options, Customize, and Unmovable Files**.
2. Specify that the *.ent, *.rst, and .key, files cannot be moved (the .41s files need not be specified).
3. Choose **File, Options, Optimization, and Save** to save the new profile.

Function Reference

When using some of CrypKey functions, you'll need background information and technical details.

T

code.

his section contains reference information that supports the other sections in this manual. It describes the functions and general error values used by CrypKey and provides assistance for programming your CrypKey code. For function declarations, refer to the sample

12.1 Function Summary

For a quick reference of function syntax, see Appendix II (Sec. 14.1: CrypKey Functions).

Table 10: Function Summary

Name	Function
AcquireLicense()	Allows an unlicensed CrypKey-protected program to acquire a license from a licensed copy. It is the opposite of the DirectTransfer() function.
CKChallenge()	Used to authenticate the CrypKey DLL, ensuring that it is not an impostor. The function name in Visual Basic is ckChallenge .
CKChallenge32()	Used to authenticate the CrypKey DLL, ensuring that it is not an impostor. The function name in Visual Basic is CKChallenge32 .
CKTimeString()	Translates, into a text string, the ULONG time values returned by the functions GetAuthorization() and FloatingLicenseSnapshot() .
CrypkeyVersion()	Returns the version number of the CrypKey library currently in use. The function name in Visual Basic is crypkeyVersion .
DirectTransfer()	Called from an application that has authorization, this function transfers the application's authorization license to another directory.
EndCrypkey()	Notifies CrypKey that the program is terminating.
ExplainErr()	Returns a text string that explains the return code of various functions.
ExplainErr2()	Returns a text string (written into string text) that explains the return code of various functions.
FloatingLicenseGetRecord()	C#.NET function only: returns a single record of the current floating license status. The function FloatingLicenseTakeSnapshot() needs to be called before this function.
FloatingLicenseSnapshot()	Returns a snapshot of all users holding or waiting for a network-floating license.
FloatingLicenseTakeSnapshot()	C#.NET function only: returns a snapshot of all users holding or waiting for a network floating license.
Get1RestInfo()	Returns authorization options, the number allowed, and the number used.
GetAuthorization()	Gets the level at which the program is authorized to run or an authorization failure code. Also used to decrement the number of uses count.

Name	Function
GetAuthorization2()	The same as GetAuthorization() , except this function decrements the number of uses count only.
GetDrivePermanentSerialData()	Proprietary CrypKey software function that accesses the following hard drive information on a customer's computer: model number, serial number, firmware data.
GetLastError()	Returns the text for the last error reported.
GetLastErrorCode()	C#.NET function only: returns the number of the last error reported. Used when you use some of the safe code alternative functions in C#.NET.
GetLevel()	Returns the level of authorization or -1 if the program is not authorized.
GetNetHandle()	Gets the network license handle of the protected program.
GetNumCopies()	Gets the number of license copies that the current site has been granted.
GetNumMultiUsers()	Gets the number of customers to whom the site license has been granted.
GetOption()	For a specific option, the function returns 1 if the option is on, 0 if the option is off, and -1 if the program is not authorized.
GetRestrictionInfo()	Gets information on any restrictions that are present in the license.
GetSiteCode()	Gets the site code for the program installation location (site).
GetSiteCode2()	Returns a pointer to the site code. The same as GetSiteCode() , except text is not written to a string.
InitCrypkey()	Initializes CrypKey with the required runtime information.
KillLicense()	Kills the protected program's license and provides a confirmation code.
ReadyToTry()	Sets up a days class, one-time automatic license without version control. The function name is readyToTry in Visual Basic.
ReadyToTryDays()	Sets up a days class, one-time automatic license. The function name is readyToTryDays in Visual Basic.
ReadyToTryRuns()	Sets up a runs class, one-time automatic license. The function name is readyToTryRuns in Visual Basic.
RegisterTransfer()	Called from an application that does not have authorization, this function registers the application by placing special files on a disk or directory.

Name	Function
SaveSiteKey()	Used to save the site key that is acquired by the customer. The key is checked before it is saved.
SetNetHandle()	Sets the network license handle reference of the protected program.
TransferIn()	Called from an application that does not have authorization, this imports an authorization license by reading special files from a disk or in a directory that were created by the TransferOut() function.
TransferOut()	Called from an application that has authorization, this exports the application's authorization license by writing special files out to a directory where a registration file was created using the RegisterTransfer() function. Only the registered application can successfully call the TransferIn() function to import this license.

12.2 General Error Values

Most CrypKey functions that return values can return general error codes in addition to function-specific error codes (described in Sec. 12.3: Function Descriptions). These general errors include network and DLL communication errors. You should always check for these return values and give some explanation to the customer if they occur—

ExplainErr() (Sec. 12.3.8) returns an appropriate message.



Note: as mentioned in the table below, errors in the –200 to –299 range result from security checks. The most common cause of these errors is the placement of the .ngn file in a different directory from the one where the executable resides. In these cases, moving the .ngn or executable to the proper location should clear the error. If it does not, please contact CrypKey technical support.

Table 11: General Errors and Solutions

Name	Value	Message	Solution
FILE_INFO_FAIL	-100	Could not get the licensing information for unknown reason (wrong operating system or insufficient memory)	If you see this error, ensure you have the most recent software. Send an email message to support@crypkey.com containing the dates of the CrypKey files you are using and we tell you if you have the most recent software.
NETWORK_DISCONNECTED	-101	The network connection appears to have been lost	If you encounter this error, ensure you are using the most recent driver. Do this by sending an email message to support@crypkey.com that states the date of the driver you are using.
NETWORK_NO_CKSERVE	-102	CKSERVE.EXE (or its equivalent) is not serving the program	The CrypKey driver you are using is not serving the directory where your CrypKey-protected application is located. All CrypKey-protected applications must be served by the driver. This error usually occurs on Windows NT. To solve this problem, ensure the NT driver is installed. If you still experience problems, send an email message containing the date of setupex.exe to support@crypkey.com . For further information about this error, see also Chapter 11, Troubleshooting, (p. 145).
NETWORK_BAD_REPLY	-103	The reply file was either damaged or tampered with	This error is usually caused by an old driver. Please contact us via email at support@crypkey.com with the date of the driver that you are using.
INIT_NOT_SUCCEEDED	-104	CrypKey initialization did not succeed	Check the return value from InitCrypKey. This should provide you with information about the

Name	Value	Message	Solution
			problem.
THUNK_FAILURE	-105	Unable to perform the thunk operation	Ensure both of the thunk DLLs (Crp95f.dll and Cryp9516f.dll) are in the application directory. If they are in the application directory and you continue to receive the error please email support@crypkey.com and report the dates of your thunk libraries.
BAD_THUNK	-106	The DLL is replaced with an impostor	If this error occurs, ensure you have the original copy of the CrypKey thunk DLLs. If you still encounter this problem, send the dates of the DLLs you are using to support@crypkey.com .
NO_MEMORY	-107	A request for memory failed	Email us at support@crypkey.com if this error occurs.
OLD_THUNK	-108	The DLL is replaced with an older version or an older version is already running	Ensure both of the thunk DLLs (Crp95f.dll and Cryp9516f.dll) are in the application directory. If they are in the application directory and you continue to receive the error please email support@crypkey.com and let us know the dates of your thunk libraries.
THUNK_BLOCKED	-109	THUNK dll is used but not released by another application	Email us at support@crypkey.com if this error occurs.
NETWORK_BAD_CRC	-110	A reply to a network request is corrupted	Email us at support@crypkey.com if this error occurs.
INIT_NOT_CALLED	-111	All functions require init to be called	Ensure InitCrypkey() is called before calling any other functions.

Name	Value	Message	Solution
EASYLIC_OPERATION_NOT_ALLOWED	-112	Certain operations such as license transfers are not allowed with EasyLicence (see Sec. 4.2.2).\	Instruct the user that this operation is not allowed.
[various messages]	-200 to -299	The .ngn file is possibly not in the same directory as the executable.	The 200-series errors are generated by security checks. Ensure that the .ngn file is in the same directory as the executable. If this action does not clear the error, email us at support@crypkey.com .

12.3 Function Descriptions

This section provides detailed information about the purpose, use, syntax, parameters, and interaction of CrypKey functions.

You can also see a summary of these functions in Sec. 12.1: Function Summary, and in Appendix II—Quick Reference (Sec. 14.1: CrypKey Functions).

Where applicable, each description includes a prototypes table, showing the following:

- C prototype, for use with the CrypKey Library
- C++ prototype, for use with the CrypKey COM object
- Visual Basic (VB) prototype, for use with the CrypKey COM object
- C#.NET prototype, for use with the CrypKey .NET assembly

Note that some of the C#.NET methods are unsafe code because they pass values by reference. These functions are provided for compatibility only. Alternate methods exist that contain safe code and are in many cases provided in this document.

12.3.1 AcquireLicense()

AcquireLicense() is a new function that allows an unlicensed CrypKey protected program to acquire a license from a licensed copy. It is the opposite of the **DirectTransfer()** function.

For this function to be successful, certain conditions must be met:

- the target directory must not already have a valid license
- the source must have at least one valid license
- if either source or target directory is not local, a CrypKey driver must be operation on the machine the directory is on

If the source license is a fixed license and has more than one copy, this function transfers exactly one copy to the target and decrement the source number of copies by one.

If the source license is a floating license with x users allowed, the entire license including x users is transferred.

Note: if the source directory is not local, the source license does not have to be a floating license to be transferred.

Usage

1. A company can distribute fixed licenses to a number of client computers by having the client acquire the license from a server blessed with multiple fixed licenses.
2. A new version of CrypKey protected software can automatically acquire its license from the old version.

Prototypes

Language	Syntax
C	<code>int AcquireLicense (char *licensePath);</code>
C++	<code>long AcquireLicense (BSTR bstrDirectory);</code>
VB	<code>AcquireLicense (strDirectory as String) as Long</code>
C#.NET	<code>CKAcquireErrorEnum AcquireLicense (String *pstrDirectory);</code>

Parameter

The license path is a pointer to a NULL terminated string that gives the directory from which the license is to be acquired. For successful operation of the function, the directory must contain a valid license for the calling program.

Return Value

If the function operates normally, it returns the value 0. Otherwise, it returns one of the error values described in the table below.

Name	Value	Message	Solution
AL_SOURCE_NOT_AUTHORIZED	-1	Source of application must be authorized for this operation	Ensure the source application has a valid license.

Name	Value	Message	Solution
AL_TARGET_APP_NOT_FOUND	-2	Target application was not found	Ensure there is the copy of the application in the target directory.
AL_FLOPPY_NOT_ALLOWED	-3	Operation not allowed on floppy disk	Do not acquire licenses from or to floppy disks.
AL_SITEKEYFILE_NOT_FOUND	-4	Source site key file was not found	Contact CrypKey technical support if you get this error.
AL_RSTKEYFILE_NOT_FOUND	-5	Source .RST file was not found	Contact CrypKey technical support if you get this error.
AL_COULD_NOT_WRITE_SITEKEYFILE	-6	Could not write to target site key file	Ensure the target directory is writable and ensure there is not a read-only copy of <appname>.key in the target directory.
AL_TARGET_WRITE_PROTECTED	-7	Target site he filed his write protected	Ensure there is not a read-only copy of <appname>.key in the target directory.
AL_SOURCE_SAME_AS_TARGET	-8	Source directory is the same has target directory	Ensure the source and target directories are different.
AL_THIS_IS_AUTHORIZED	-9	Target already has authorization	Remove the current authorization using the KillLicense function if you are sure you don't want the current license.
AL_LICENSE_SOURCE_NOT_FOUND	-10	Source of application was not found	Ensure the application is in the source directory.

Name	Value	Message	Solution
AL_COULD_NOT_INIT IALIZE_SOURCE	-11	Could not initialize CrypKey in source directory	A driver is probably missing. Contact CrypKey technical support if you get this error.

12.3.2 CKChallenge()

Note: it is recommended that, instead of this function, 32-bit programs use **CKChallenge32()**—see Sec. 12.3.3.

CKChallenge() verifies that the CrypKey DLL has not been replaced with an impostor DLL (it also verifies the authenticity of the CrypKey external program. In order to use this function, you must have the company and password numbers that CrypKey gave you when you received your Master Key and User Key.

If you do a calculation with these and two other randomly chosen numbers and then call the DLL to do the same calculation with the same four numbers, the DLL should arrive at the same answer. Only the real DLL can decode the Master Key and User Key to get your company and password numbers for the calculation, so an impostor DLL has great difficulty providing the correct answer for any possible values of the two random numbers.

Usage

Call **CKChallenge()** when the program first starts to ensure the authenticity of the DLL (or the external program if you are using external linking). You can also call this function periodically to check that the license is valid instead of (or in addition to) using **GetAuthorization()**. If the license is not valid, **CKChallenge()** does not return the correct codes.

Prototypes

Language	Syntax
C	unsigned short ckChallenge (unsigned short random1, unsigned short random2);
C++	long CKChallenge (long nRandom1, long nRandom2);

Language	Syntax
VB	CKChallenge (nRandom1 as Long, nRandom2 as Long) as Long
C#.NET	int CKChallenge (int nRandom1, int nRandom2);

Parameters

Random1

Random1 is a 16-bit number chosen at random. A good way to get a random number is to use the system time. This number must be less than 32,768.

Random2

Random2 is a 16-bit number chosen at random. A good way to get a random number is to use the system time. This number must be less than 32,768.

Return Value

CKChallenge() returns the result of the challenge calculation done using the two parameters of this function. The actual calculation that the function returns is:

```
initialize result to 0
repeat 11 times:
    result = (result * random1 + companyNum) mod 16381 +
    (result * random2 + passNum) mod 16369
```

where:

- result is the result of the calculation
- random1 and random2 are two numbers between but not equal to 0 and 65,536.
- companyNum and passNum are your CrypKey-assigned secret numbers

The actual calculation that you would do in C code is as follows:

```
#define UINT unsigned integer
#define ULONG unsigned long
#define USHORT unsigned short
```



```

USHORT Challenge(ULONG companyNum, ULONG passNum, USHORT
random1,
USHORT random2)
{
    int I;
    ULONG ret,r1,r2;
    r1 = (ULONG)random1;
    r2 = (ULONG)random2;
    ret = 0;
    for (i=0,i<11;i++)
        ret = (ret*r1+companyNum)%16381L+(ret*r2+passNum)%16369L;
    return (USHORT)ret;
}

```

Notes

The **InitCrypkey()** and **GetAuthorization()** functions must have been called and returned successfully for **CKChallenge()** to work. Note that **CKChallenge()** does not return the correct code if the program is not authorized. This gives you another way (besides **GetAuthorization()**) to check for a valid license. This function can only be called from applications that have their own Master Key and User Key. **CKChallenge()** does not work if you are using the Master Key and User Key for example.exe.

12.3.3 CKChallenge32()

CKChallenge32() is used to authenticate the CrypKey DLL, ensuring that it is not an impostor. It is also used to verify the authenticity of the CrypKey external program. In order to use this function, you must have the company and password numbers provided by CrypKey with the Master Key and User Key.

If you perform a calculation with these and two other randomly chosen numbers and then call the DLL to do the same calculation with the same four numbers, the DLL should arrive at the same answer. Only the real DLL can decode the Master Key and User Key to get your company and password numbers for the calculation. An impostor DLL has great difficulty providing the correct answer for any possible values of the two random numbers.

Usage

Call **CKChallenge32()** when the program starts to ensure the authenticity of the DLL (or the external program if you are using external linking). You can also call this function periodically to check that the license is valid instead of (or in addition to) using **GetAuthorization()**. If the license is not valid, **CKChallenge32()** will not return the correct codes.

Prototypes

Language	Syntax
C	Long CKChallenge32 (long random1, long random2);
C++	Long CKChallenge32 (long nRandom1, long nRandom2);
VB	CKChallenge32 (nRandom1 as Long, nRandom2 as Long) as Long
C#.NET	int CKChallenge32 (int nRandom1, int nRandom2);

Parameters

Random1

Random1 is a 31-bit number chosen at random. A good way to get a random number is to use the system time. This number must be less than $2^{31}-1$.

Random2

Random2 is a 31-bit number chosen at random. A good way to get a random number is to use the system time. This number must be less than $2^{31}-1$.

Return Value

CKChallenge32() returns the result of the challenge calculation done using the two parameters of this function. The actual calculation that the function returns is:

```
{
int i;
long ret;
```

```

ret = 1;
for(i=2;i<11;i++)
{
    ret = ret%32769 * ( (random1/i)%32769 + (companyNum/i)%32769);
    ret = ret%32769 * ( (random2/i)%32769 + (passNumDiv2/i)%32769);
}

return(ret);
}

```

where:

- ret is the result of the calculation.
- random1 and random2 are two numbers between but not equal to 0 and $2^{31}-1$.
- companyNum and passNum are your CrypKey-assigned secret numbers.

The actual calculation that you would do in C code is also shown in **example.c** in the Sample directory.

Notes

The **InitCrypkey()** and **GetAuthorization()** functions must be called and returned successfully for **CKChallenge32()** to work. Note that **CKChallenge32()** does not return the correct code if the program is not authorized—this gives you another way (besides **GetAuthorization()**) to check for a valid license. This function can only be called from applications that have their own Master Key and User Key.

12.3.4 CKTimeString()

The functions **GetAuthorization()** and **FloatingLicenseSnapshot()** both return ULONG values that represent time. This function translates the time into a text string.

See **example.c** for an example of how to use this function.

Prototype

```
char * FUNCTYPE CKTimeString(char *timestringbuf, ULONG t)
```

Parameters

timestringbuf

timestringbuf is a buffer to put the time string at least 30 bytes wide. The time string is in the form YY-MM-DD HH:MM:SS.

t

t is the 32-bit integer value received from the CrypKey function to be translated to text. It returns a pointer to *timestringbuf* for convenience.

12.3.5 CrypkeyVersion()

CrypkeyVersion() returns the version number of the CrypKey system currently in use.

Usage

Call **CrypkeyVersion()** whenever you need to know which version of CrypKey is linked to your program code.

Prototypes

Language	Syntax
C	<code>int CrypkeyVersion ();</code>
C++	<code>long CrypKeyVersion ();</code>
VB	<code>CrypKeyVersion () as Long</code>
C#.NET	<code>int CrypKeyVersion ();</code>

Return Value

CrypkeyVersion() returns a two-digit number that indicates which version of the CrypKey library is in use (33 indicates v3.3, 40 indicates v4.0, 43 indicates v4.3, etc.).

12.3.6 DirectTransfer()

Called from an application that has authorization, **DirectTransfer()** transfers the application's authorization license to another directory. This transfer could be from one directory to another on the same computer or from one network drive to another.

The directory where the function is initiated must also be the directory where the license resides.

Note: The **InitCrypKey()** and **GetAuthorization()** functions must be successfully called before **DirectTransfer()** is used.

Usage

Use **DirectTransfer()** whenever you need to transfer a license from one location to another and there is a direct connection between the two drives.

Prototypes

Language	Syntax
C	<code>int DirectTransfer (char *directory);</code>
C++	<code>long DirectTransfer (BSTR bstrDirectory);</code>
VB	<code>DirectTransfer (strDirectory as String) as Long</code>
C#.NET	<code>CKDirectErrorEnum DirectTransfer (String *pstrDirectory);</code>

Parameter

Directory

Directory is a pointer to a null-terminated text string that contains the path of the directory to which to export the license.

Return Value

Table 12: DirectTransfer() Values

Name	Value	Message	Solution
DT_OK	0	The function completed successfully	
DT_THIS_NOT_AUTHORIZED	-1	This site is not authorized; there is nothing to transfer	If there is no authorization, you cannot transfer a license. Obtain authorization before attempting to transfer your license.
DT_TARGET_APP_NOT_FOUND	-2	A copy of the program must be in the target directory	A copy of the program must be located in the target directory. If it is not there, you cannot transfer the license.
DT_FLOPPY_NOT_ALLOWED	-3	Illegal attempt to transfer license to a target directory on a floppy.	There are certain steps that you must take when transferring by floppy disk. If you are using this call, ensure you are doing a direct transfer, not a transfer by floppy disk.
DT_SITEKEYFILE_NOT_FOUND	-4	The Site Key file could not be opened.	This function must have access to the site key file. If this function cannot access the site key file, the function gives you errors. Ensure the file is in the application directory. If you encounter this problem, please email support@crypkey.com .

Name	Value	Message	Solution
DT_RSTKEYFILE_NO T_FOUND	-5	The restriction file could not be opened.	This function must have access to the restriction file. If this function cannot access the restriction file, the function gives you errors. Ensure this file is in the application directory. If you encounter this problem, please email support@crypkey.com .
DT_COULDNOT_WRI TE_SITEKEYFILE	-6	The target directory may be write protected.	This function must read and write to specific files in your directory. If you are able to do this but still receive this error, please email us at support@crypkey.com .
DT_SOURCE_WRITE_ PROTECTED	-7	The directory of the source license for the transfer is write protected.	This function must read and write in both the target and source directories. If you are able to do this but continue to have problems, please email us at support@crypkey.com .
DT_SOURCE_SAME_ AS_TARGET	-8	The source directory is also being used as the target directory.	To use this function you must be transferring from one directory to another. You cannot transfer a license from a directory into the same directory.

General error values also apply — see Section 12.2: General Error Values.

Notes

The **DirectTransfer()** function overwrites an existing license at the target location without warning. Therefore, always be careful to check whether overwriting is desirable.

12.3.7 EndCrypkey()

EndCrypkey() notifies CrypKey that the protected program is being terminated. This allows CrypKey to ensure that its state of operation is released properly beforehand.

Usage

Ensure that **EndCrypkey()** is called before the protected program terminates. For example, if you are writing in C++, a good place to call this function is in the application object destructor. Note that you must ensure **EndCrypkey()** is called no matter how the customer shuts down your program, even if the customer double-clicks on the System menu.

Prototypes

Language	Syntax
C	<code>void EndCrypkey();</code>
C++	<code>void EndCrypKey();</code>
VB	<code>EndCrypKey()</code>
C#.NET	<code>void EndCrypKey();</code>

Notes

EndCrypkey() is used only to update the status of a multi-user floating license. It is not required for any other purpose; therefore, if a program terminates abnormally without calling this function, there will not be any abnormal effects from CrypKey when the program is restarted.

12.3.8 ExplainErr()

ExplainErr() returns a string explaining the return codes of CrypKey functions.

Note: the data type CKExplainEnum (see Sec. 5.4.7: COM Object Constants) contains the constant values for errors handled by COM Objects. See also the **GetLastError()** function.

Usage

It is not mandatory that you show these particular explanations to the user, but it is recommended that some explanation be given when a CrypKey error does occur.

Prototypes

Language	Syntax
C	<code>char *ExplainErr (int functioncode, int errcode);</code>
C++	<code>BSTR *ExplainErr (long nFunctionCode, long nErrorCode);</code>
VB	<code>ExplainErr (nFunctionCode as Long, nErrorCode as Long) as ByRef String</code>
C#.NET	<code>String *ExplainErr (CKExplainEnum nFunctionCode, int nErrorCode);</code>

Parameters

Functioncode

Functioncode must be one of the elements in the table below. These codes are defined in `crypkey.h`. *Functioncode* determines which function is explained.

Table 13: Functioncode Values

Name	Value	Function
EXP_AUTH_ERR	1	GetAuthorization()
EXP_GET_SITECODE_ERR	2	GetSizeCode()
EXP_SAVE_SITEKEY_ERR	3	SaveSiteKey()
EXP_REG_ERR	4	RegisterTransfer()
EXP_TO_ERR	5	TransferOut()
EXP_TI_ERR	6	TransferIn()
EXP_DT_ERR	7	DirectTransfer()
EXP_INIT_ERR	8	InitCrypkey()
EXP_RTT_ERR	9	ReadyToTry(),...Days(), ...Runs()
EXP_KL_ERR	10	KillLicense()

Errcode

Errcode is the specific error code returned by one of the SLAPI functions. The functions are listed in Table 13.

Return Value

ExplainErr() returns the appropriate error message.

12.3.9 ExplainErr2()

ExplainErr2() returns a string explaining the return codes of various other SLAPI functions. This function is similar to **ExplainErr()**, but the error message is written into your string text instead of being returned as a string.

Note: the data type CKExplainEnum (see Sec. 5.4.7: COM Object Constants) contains the constant values for errors handled by COM Objects. See also the **GetLastError()** function.

Usage

It is not mandatory that you show these particular explanations to the user, but it is recommended that some explanation be given when a CrypKey error does occur.

Prototypes

Language	Syntax
C	<code>void ExplainErr2 (int func, int err, char *text);</code>
C++	<code>void ExplainErr2 (long nFunctionCode, long nErrorCode, BSTR *pbstrError);</code>
VB	<code>ExplainErr2 (nFunctionCode as Long, nErrorCode as Long, strError as ByRef String)</code>
C#.NET	<code>String *ExplainErr2 (CKExplainEnum nFunctionCode, int nErrorCode);</code>

Parameters

Functioncode

Functioncode must be one of the elements in the table below. These codes are defined in `cryptkey.h`. *Functioncode* determines which function is explained.

Table 14: Functioncode Values

Name	Value	Function
EXP_AUTH_ERR	1	GetAuthorization()
EXP_GET_SITECODE_ERR	2	GetSizeCode()
EXP_SAVE_SITEKEY_ERR	3	SaveSiteKey()
EXP_REG_ERR	4	RegisterTransfer()
EXP_TO_ERR	5	TransferOut()
EXP_TI_ERR	6	TransferIn()
EXP_DT_ERR	7	DirectTransfer()
EXP_INIT_ERR	8	InitCrypkey()
EXP_RTT_ERR	9	ReadyToTry(), ...Days(), ...Runs()
EXP_KL_ERR	10	KillLicense()
EXP_AL_ERR	11	AcquireLicense()
EXP_HDSN_ERR	12	GetDrivePermanentSerialData()
EXP_FLS_ERR	13	FloatingLicenseSnapshot()

Errcode

Errcode is the specific error code returned by one of the SLAPI functions. The functions are listed in Table 14.

Text

The *text* parameter is a pointer to the buffer where the error message is written. It must be 80 characters in size.

Note

ExplainErr2() is an optional alternative to the **ExplainErr()** function.

12.3.10 FloatingLicenseGetRecord

This function is used only in COM Objects and C#.NET Assemblies. Called from an application that has authorization, the function returns a single record of the current floating license status. The function **FloatingLicenseTakeSnapshot ()**, used only in COM Objects and C#.NET Assemblies, must be called before this function.

Prototypes

Language	Syntax
C	[n/a]
C++	long FloatingLicenseGetRecord (long nRecord, IcrypKeySDKFloatRecord **ppRecord, long *pnResult);
VB	[n/a]
C#.NET	int FloatingLicenseGetRecord (int nRecord, CrypKeyFloatRecord *pRecord);

Notes

This function operates only in COM Objects and C#.NET Assemblies.

12.3.11 FloatingLicenseSnapshot

The **FloatingLicenseSnapshot()** function gives a snapshot of all users holding or waiting for a network floating license. The function fills the buffer with entries, one for each user accessing the license.

Each entry contains the computer name, user login name, local time at which the user started using or waiting for a license, and the user’s queue status (0 means they have a license, otherwise it is an integer that indicates the order in which each entry gets a license). This is the pertinent information given; the rest is for internal use.

Prototypes

Language	Syntax
C	<code>int FloatingLicenseSnapshot (unsigned long bufSize, int *numEntries, FLS_REC *buf);</code>
C++	<code>long FloatingLicenseTakeSnapshot (long *pnEntries, long *pnResult);</code>
VB	<code>FloatingLicenseTakeSnapshot (pnEntries as ByRef Long) as Long</code>
C#.NET	<code>CKFloatSnapErrorEnum FloatingLicenseTakeSnapshot (int *pnEntries);</code> <code>// unsafe</code> <code>- or -</code> <code>FloatSnapErrorEnum FloatingLicenseTakeSnapshot2 (); // safe</code> <code>- plus -</code> <code>int FloatingLicenseGetNumEntries (); // safe</code>

Parameters

Bufsize

Bufsize is the size of the buffer allocated to receive the information.

NumEntries

NumEntries is a pointer to an integer-sized buffer that receives the number of entries in the table.

Buf

Buf is a pointer to a buffer of at least *bufsize* bytes in size and receives one entry for each user that is currently accessing, or attempting to access, the license.

Return Value**Table 15: Floating License Snapshot – Return Values**

Name	Value	Meaning	Solution
FLS_OK	0	The function completed successfully.	
FLS_NO_NETWORK_LICENSE	-1	Application does not have a floating license.	Ensure you have a floating license before using this function.
FLS_NETWORK_TABLE_NOT_FOUND	-2	Network table file (.tb2) could not be found.	Contact CrypKey technical support if you get this error.
FLS_COULDNOT_OPEN_USERTABLE_FILE	-3	Network table file (.tb2) could not be opened.	Contact CrypKey technical support if you get this error.
FLS_BUFFER_TOO_SMALL	-4	Buffer not large enough to hold snapshot information.	Ensure you have allocated enough space in your buffer to receive this information.

Notes

The version of this function used in COM Object and C#.NET is named **FloatingLicenseTakeSnapshot()** and is used in conjunction with the COM Object and C#.NET function **FloatingLicenseGetRecord()**.

As commented in the prototypes, the first C#.NET code given:

```
CKFloatSnapErrorEnum FloatingLicenseTakeSnapshot (long
*pnEntries);
```

is unsafe code.

The second C#.NET code given:

```
FloatSnapErrorEnum FloatingLicenseTakeSnapshot2 (); safe
- plus -
long FloatingLicenseGetNumEntries ();
```

is safe code.

FLS_REC is the structure of each informational entry. It is defined in CrypKey.h. The contents of each record are described in Table 16: Floating License Snapshot – Record Structure. See also Sec. 5.4.9: Referencing the Floating License Snapshot Record, where you will find the C++ and C#.NET prototypes for functions used to encapsulate the FLS record.

Table 16: Floating License Snapshot – Record Structure

Item Name	Size (Bytes)	Description
Id	2	unique id for the record, used internally
Update	4	timestamp of last update, used internally
Status	2	<ul style="list-style-type: none"> if running: 0 if waiting: a positive integer representing the queue position
Starttime	4	local timestamp of when the application started running or waiting in the queue — could be seconds since 1900 or 1970, depending on the compiler
UserName	16	text user name
ComputerName	16	text computer name
Spare	32	reserved

12.3.12 Get1RestInfo()

Get1RestInfo() is used to return one of three different types of restriction information: license type, number of days or runs allowed, or number of days or runs used. The type of information returned depends on the value of the *which* input parameter.

Usage

Use **Get1RestInfo()** whenever you need to get license restriction information.

Prototypes

Language	Syntax
C	<code>int Get1RestInfo (int which);</code>
C++	<code>long Get1RestInfo (long nWhich);</code>
VB	<code>Get1RestInfo (nWhich as Long) as Long</code>
C#.NET	<code>CKGet1ErrorEnum Get1RestInfo (CKWhichEnum nWhich);</code>

Parameter

Which

The *which* parameter is set to one of the following:

- 1: returns *authopt* (indicating whether the license is unlimited, days-based, or runs-based)
- 2: returns *num_allowed* (indicating the number of days or runs allowed)
- 3: returns *num_used* (indicating the number of days or runs used)

Return Value

Table 17: Get1RestInfo() Values

Name	Value	Message
not defined	0-32768	Returns the <i>which</i> variable as defined above.
G1_OUT_OF_RANGE	-1	The <i>which</i> variable is out of range.

Note

Get1RestInfo() is an optional alternative to using the **GetRestrictionInfo()** function.

12.3.13 GetAuthorization()

GetAuthorization() gets the level at which the protected program is authorized to run along with the options enabled or an authorization failure code. It is also used to decrement the number of runs, if the application is restricted by runs.

Usage

GetAuthorization() can be called at your discretion to determine the level of authorization that the user is granted. It can also be called to decrement the number of uses the program has left if it is restricted by number of runs. Call this function after calling any function that changes the license state, such as **SaveSiteKey()**, **TransferOut()** or **TransferIn()**, so that the program can alter its behavior according to the change that has occurred.

Prototypes

Language	Syntax
C	<code>int GetAuthorization (unsigned long *oplevel, int decrement);</code>
C++	<code>long GetAuthorization (long *pnOptLevel, long nDecrement);</code>
VB	<code>GetAuthorization (nOptLevel as ByRef Long, nDecrement as Long) as Long</code>
C#.NET	<code>CKAuthErrorEnum GetAuthorization (int *pnOptLevel, int nDecrement); // unsafe</code>

Parameters

Oplevel

Oplevel is a pointer to a 32-bit buffer. The *oplevel* is returned in this variable if the program is authorized. It is a combination of option and level information stored in 32 bits.

Decrement

If this number is not zero and the protected program is restricted to the number of runs allowed, the number of runs remaining decrements by this number.

Return Value

Table 18: GetAuthorization() Values

Name	Value	Meaning	Solution
not defined	>0	The program is authorized to run. The program has a floating license and more users are requesting to use the program than the license allows. The number indicates how many users must quit before the next user is authorized to run. CrypKey queues requests for access and provides access on a first-come, first-served basis.	You may want your program to allow the user to wait, by recalling this function every 30 seconds until a seat becomes available, or the user gives up.
AUTH_OK	0	The program is authorized to run.	
AUTH_INIT_FAIL	-1	CrypKey has not been initialized.	If you get this error message, please obtain the return value from InitCrypkey() . Something has gone wrong in the InitCrypkey() function call.

Name	Value	Meaning	Solution
AUTH_DISALLOW_FLOPPY	-2	An attempt is being made to run the protected program from a floppy disk.	If this error occurs, you are probably trying to run a protected program from a floppy disk. CrypKey must write a signature to the hard drive. If the program is running from a floppy disk, it cannot write to the hard drive.
AUTH_BAD_PATH	-3	Could not read the site key file.	If this error occurs, there is a problem reading the site key file on your hard drive. You must have the ability to read and write directly to the hard drive. If you are able to read and write directly to the hard drive and you continue to receive the error, please email support@crypkey.com .
AUTH_NOT_PRESENT	-4	The program has never been authorized at the current site.	If you receive this error, the program needs to be authorized. You can authorize your program using your Site Key Generator.
AUTH_DIFFERENT	-5	The program is moved or the password is incorrect.	If you receive this error, ensure the User Key you are using in the InitCrypkey() function call is the same as the User Key that was given to you by CrypKey. If you continue to receive this error, please email the User Key to support@crypkey.com .

Name	Value	Meaning	Solution
AUTH_BAD_MASTERKEY	-6	The Master Key is not correct for this program.	If you receive this error, ensure the Master Key you are using matches the program name that you are using in the InitCrypkey() function call. The file name that must be in the InitCrypKey() function call is the license file name that you gave to CrypKey. If you continue to have problems, please email all of the information you have to support@crypkey.com .
AUTH_SITEKEY_CRC	-7	The site key file is damaged or tampered with.	If you receive this error, delete the license files and run the application again. You must be reauthorized at this time. Please email support@crypkey.com if this solution does not work for you.
AUTH_TIME_TO_O_EARLY	-8	The system time is set to a date earlier than the date on which the Site Key was issued.	This error occurs when the system time is moved. You must re-authorize the software. The system time cannot be moved by more than one hour and fifteen minutes at any time.
AUTH_TIME_SETBACK	-9	The program is restricted by days and the system time is set back.	This error occurs when the system time is moved. You must re-authorize the software. The system time cannot be moved by more than one hour and fifteen minutes at any time.

Name	Value	Meaning	Solution
AUTH_TIME_RUNOUT	-10	The program is restricted by days and the time has expired.	The user's time has expired. The user must be re-authorized to use the software.
AUTH_RUNS_RESTR	-11	The program is restricted by runs and the number allowed is reached.	The user's time has expired. The user must be re-authorized to use the software.
AUTH_NOT_ENOUGH_RUNS	-12	The user tried to perform an operation that requires more runs than are available.	Ensure the user has more runs left than the number stated in the decrement parameter of this function.
AUTH_MISSING_RST_FILE	-13	The restriction file is deleted.	The CrypKey-protected application cannot find the .rst file in the application directory. If this file is moved or deleted, the protected application will not run. The only way to solve this problem is to re-authorize the user.
AUTH_RST_BAD_CRC	-14	The restriction file is damaged or tampered with.	This error occurs if the user has attempted to alter the .rst file. The only solution to this problem is to re-authorize the user. Please contact support@crypkey.com if you continue to receive this error.

Name	Value	Meaning	Solution
AUTH_RST_BAD_LOCATION	-15	The restriction file is moved or overwritten.	If this error occurs, you must re-authorize the user. This error can occur for several reasons (e.g., the user runs Norton Utilities Speed Disk, the user uninstalls and reinstalls, the user changes the time on the PC, the user moves the application directory, etc.). If you continue to receive this error, please contact support@crypkey.com .
AUTH_ENTRY_CHECK_FAIL	-16	The user has attempted to reuse an old site key.	This error is created when a user is trying to enter a site key that was not created for the site code the user has. Please obtain the site code and request the user to enter the new Site Key.

Name	Value	Meaning	Solution
AUTH_NETTABLE_FILE_FAIL	-17	Could not open or read the network user table file.	If this error occurs, ensure the user is able to read and write to the hard drive. When running over a network, CrypKey creates a file with the extension <code>._tb</code> . This file should be opened and closed every time you run a CrypKey-protected application over the network. If you continue to receive this error, please contact support@crypkey.com .
AUTH_NETMAX_EXCEEDED	-18	More people are using the protected program than SLAPI can handle.	This error occurs when more users try to use the protected application than authorized. If you cannot eliminate this error, please contact support@crypkey.com .
AUTH_NETWORK_NOT_ALLOWED	-19	An attempt is being made to run the program in client/server mode and a floating license has not been issued for the site.	This error occurs when the user who is trying to run the application has not received a site key that has network privileges. If you continue to receive this error after granting the permissions, please contact support@crypkey.com .
AUTH_RSTFILE_WRITE_PROTECTED	-20	.RST file is write protected	Ensure this file is not write protected.
AUTH_TIME_CLOCK_TAMPERING	-21	Computer clock tampering detected	Reboot your computer. Contact support if this error persists.
KEY_BAD_LOCATION	-22	Application site key is moved or copied	If you have not moved or copied your application for this file, contact support.

General error values also apply—see Section 12.2: General Error Values.

Notes

The options are single bits of information that can be turned on or off individually when a license is created using the Site Key Generator. The options exist so that you can turn on or off different features of your product. The options are stored in a numerical sequence starting at one with the high-end bit of the *oplevel* 32-bit value. The remaining bits are used to hold the level number.

The level is a number that is set by the Site Key Generator. It incrementally controls the quality of some services your software provides, such as the number of records allowed for a database. The maximum level depends on the number of options you have configured.

There are 32 bits available to both the option and level values. For example, if four options are defined in *skw.ini*, *oplevel* looks like this:

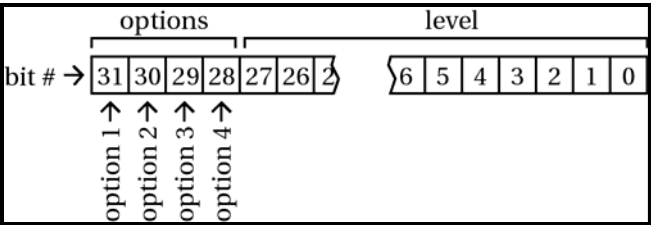


Figure 56: Options and Levels

In this example, the maximum level is 2^{28} or 268,435,456.

InitCrypkey() must be called at least once by your application before using the **GetAuthorization()** function, and this function must be called at least once before any other SLAPI function is called.

The options or levels can also be obtained by using **GetOption()** or **GetLevel()**. For security reasons, this is the only function, along with **GetAuthorization2()**, that can tell you if an incorrect password is entered into the Site Key Generator.

The C#.NET prototype for the **GetAuthorization()** function is unsafe code. The safe C#.NET alternative is **GetAuthorization2()**.

12.3.14 GetAuthorization2()

This function is similar to **GetAuthorization()** in that it determines if the program is authorized to run; and decrements the license count. However, **GetAuthorization2()** does not obtain option and level information.

Usage

This function is used in the same manner as **GetAuthorization()** and is intended for languages that are unable to obtain the *oplevel* data or for cases where you do not want to obtain the *oplevel* data.

Prototypes

Language	Syntax
C	<code>int GetAuthorization2 (int decrement);</code>
C++	<code>long GetAuthorization2 (long nDecrement);</code>
VB	<code>GetAuthorization2 (nDecrement as Long) as Long</code>
C#.NET	<code>CKAuthErrorEnum GetAuthorization2 (int nDecrement);</code>

Parameter

Decrement

If the *decrement* number is not zero and the protected program is restricted by the number of runs allowed, the number of runs remaining are decremented by the *decrement* number.

Return Value

See **GetAuthorization()** for a description of the return values for this function.

Notes

InitCrypkey() must be called at least once by your application before the **GetAuthorization2()** function is used. **GetAuthorization2()** must be called at least once before any other SLAPI function is called.

For security reasons, this is the only function, along with **GetAuthorization()**, that can tell you if an incorrect password is entered into the Site Key Generator.

This function is an optional alternative to using the **GetAuthorization()** function.

12.3.15 GetDrivePermanentSerialData()

GetDrivePermanentSerialData() is a proprietary CrypKey software function that accesses the following hard drive information on a customer’s computer: model number, serial number, firmware data.

12.3.16 GetLastError()

This function returns the text for the last error. Unlike the SDK functions **ExplainErr()** and **ExplainErr2()**, the function code does not need to be supplied. The function code is set each time one of the functions in the COM object is called. Thus, this function returned only the error text for the last function called.

Prototypes

Language	Syntax
C	[n/a]
C++	BSTR *GetLastError ();
VB	GetLastError () as String
C#.NET	String *GetLastError ();

Notes

See also **GetLastErrorCode()**.

12.3.17 GetLastErrorCode()

This function, used only in C#.NET Assemblies, returns **only the number** for the last error. Unlike the SDK functions **ExplainErr()** and **ExplainErr2()**, the function code does not need to be supplied. The function code is set each time one of the functions in the COM object is called. Thus, this function returns only the error number for the last function called.

Prototypes

Language	Syntax
C	[n/a]
C++	[n/a]
VB	[n/a]
C#.NET	<code>int GetLastError();</code>

12.3.18 GetLevel()

This function returns the level at which the protected program is authorized to run.

Usage

You may use this function at your discretion.

Prototypes

Language	Syntax
C	<code>unsigned long GetLevel (int numDefineOpts);</code>
C++	<code>long GetLevel (long nNumOptions);</code>
VB	<code>GetLevel (nNumOptions as Long) as Long</code>
C#.NET	<code>int GetLevel (long nNumberOfOptions);</code>

Parameter

NumDefineOpts

NumDefineOpts is the number of options that the protected program has defined.

Return Value

The return value is the current level that is available to the user from the program.

12.3.19 GetNetHandle()

The **GetNetHandle()** function gets and returns the network license handle of the protected program.

Usage

If you want several SLAPI-licensed programs sharing a single network seat, you can use this function to get the handle and then distribute it using the **SetNetHandle()** function. This function should be called only after successful calls to **InitCrypkey()** and **GetAuthorization()** have been made.

Prototypes

Language	Syntax
C	<code>unsigned short GetNetHandle ();</code>
C++	<code>long GetNetHandle ();</code>
VB	<code>GetNetHandle () as Long</code>
C#.NET	<code>int GetNetHandle ();</code>

Return Value

This function returns the handle to a network resource.

Notes

The user must design a way to transmit the return value to other executables sharing the network seat (see the **SetNetHandle()** function description in Section 12.3.33: **SetNetHandle()** for more information.

Only one of the programs sharing the network seat can call **EndCrypkey()** successfully—preferably, the last program to stop executing calls this function since this means that usage of the network seat is clearly no longer required.

12.3.20 GetNumCopies()

GetNumCopies() obtains the number of copies that the site is granted under the current license provided to it.

Usage

If you want to grant multiple copy site licenses, the program should display the number of copies it is granted to the user. The copies can be distributed by the user through the transfer functions.

Prototypes

Language	Syntax
C	<code>int GetNumCopies ();</code>
C++	<code>Long GetNumCopies ();</code>
VB	<code>GetNumCopies () as Long</code>
C#.NET	<code>CKCopiesErrorEnum GetNumCopies ();</code>

Return Value

Table 19: GetNumCopies() Values

Name	Value	Message
Not defined	0-255	Returns the number of license copies available.
GNC_CRYPTKEY_NOT_INITIALIZED	-1	CrypKey has not been initialized.

12.3.21 GetNumMultiUsers()

GetNumMultiUsers() gets the number of users that the site has granted in its license.

Usage

If you want to grant multiple-user site licenses, the application should display the number of users it has granted to the user.

Prototypes

Language	Syntax
C	<code>int GetNumMultiUsers ();</code>
C++	<code>long GetNumMultiUsers ();</code>
VB	<code>GetNumMultiUsers () as Long</code>
C#.NET	<code>int GetNumMultiUsers ();</code>

Return Value

This function returns the number of concurrent network uses granted in the license.

12.3.22 GetOption()

GetOption() determines the status of a specific option. The function gets the option bit specified in the current authorization license.

Usage

You can call this function at your discretion.

Prototypes

Language	Syntax
C	<code>int GetOption (int numDefineOpts, int optnum);</code>
C++	<code>long GetOption (long nNumOptions, long nOptionNum);</code>
VB	<code>GetOption (nNumOptions as Long, nOptionNum as Long) as Long</code>
C#.NET	<code>CKOptionErrorEnum GetOption (int nNumberOfOptions, int nOptionNumber);</code>

Parameters

NumDefineOpts

NumDefineOpts is the number of options that the protected program has.

Optnum

Optnum is the number of the option for which you want a status.

Return Value

Table 20: GetOption() Values

Name	Value	Message
OPTION_ON	1	The specified option is on.
OPTION_OFF	0	The specified option is off.
OPTION_UNAUTHORIZED	-1	The program is not authorized.

Note

This function is a simpler alternate to obtaining the option from the information passed by the **GetAuthorization()** function.

12.3.23 GetRestrictionInfo()

GetRestrictionInfo() gets information on any restrictions that are present in the license.

Usage

It is not necessary to take any action on this information, however, you may want to display this information at your discretion.

Prototypes

Language	Syntax
C	<code>int GetRestrictionInfo (int *authopt, unsigned long *start_date, int *num_allowed, int *num_used);</code>
C++	<code>long GetRestrictionInfo (long *pnAuthOpt, long *pnStartDate, long *pnNumAllowed, long *pnNumUsed);</code>
VB	<code>GetRestrictionInfo (nAuthOpt as ByRef Long, nStartDate as ByRef Long, nNumAllowed as ByRef Long, nNumUsed as ByRef Long) as Long</code>
C#.NET	<code>CKRestrictionErrorEnum GetRestrictionInfo (int *pnAuthOpt, int *pnStartDate, int</code>

	<code>*pnNumberAllowed, int *pnNumberUsed);</code>
--	--

Parameters

Authopt

Authopt is set to one of the values listed in the following table (as defined in `cryptkey.h`):

Table 21: Authopt Values

Name	Value	Meaning
RESTR_NONE	0	Unlimited
RESTR_TIME	1	Days
RESTR_RUNS	2	Runs

Start_date

Start_date is set to the date the license was issued, in seconds since the start of the year 1970 (Universal Coordinated Time).

Num_allowed

Num_allowed is set to the number of days or runs allowed by the current license.

Num_used

Num_used is set to the number of days or runs in the current license used up so far.

Return Value

Table 22: Num_used Values

Name	Value	Meaning
GRI_OK	0	The restriction information is valid.
GRI_INVALID	-1	The restriction information is not valid.

Notes

The **GetRestrictionInfo()** function is valid only if the protected program is authorized to run or if the program is not authorized to run due to expired restrictions (i.e., if the days or runs have expired).

The algorithm for breaking down the `start_date` for display in C code is as follows:

```
#include "time.h"
char* _sgtime(long ltime);
m_datetime = _sgtime(start_date);
char* _sgtime(long ltime)
{
    static char time_str[60];
    struct tm *t;
    ltime -= 2209075200;
    t = gmtime(&ltime);
    if(t != NULL)
    {
        strcpy(time_str, _ultoa(t->tm_mday, text, 10));
        strcat(time_str, "-");
        strcat(time_str, _ultoa(t->tm_mon+1, text, 10));
        strcat(time_str, "-");
        strcat(time_str, _ultoa(t->tm_year, text, 10));
        strcat(time_str, " ");
        strcat(time_str, _ultoa(t->tm_hour, text, 10));
        strcat(time_str, ":");
        strcat(time_str, _ultoa(t->tm_min, text, 10));
        strcat(time_str, ":");
        strcat(time_str, _ultoa(t->tm_sec, text, 10));
    }
    else
    {
        strcpy(time_str, "Predates 01-01-70 00:00:00");
    }
    return(time_str);
}
```

In Visual Basic, you can pass the value of *start_date* returned by **GetRestrictionInfo()** to a single precision variable and then add 2^{16} (65,536) to the result to get the correct number. This, in effect, replaces the bit that was truncated because Visual Basic's long integer is signed, but the value that is returned is unsigned.

12.3.24 GetSiteCode()

GetSiteCode() obtains the site code for the computer and directory from which a protected program is run. The site code must be reported by the user before a site key can be issued for the program.

Usage

If the protected program detects it is not authorized via the **GetAuthorization()** function, it must, at minimum, show the program's site code and allow the user to enter a site key.

Prototypes

Language	Syntax
C	<code>int GetSiteCode (char *site_code);</code>
C++	<code>long GetSiteCode (BSTR *pbstrSiteCode);</code>
VB	<code>GetSiteCode (strSiteCode as ByRef String) as Long</code>
C#.NET	<code>CKCodeErrorEnum GetSiteCode (StringBuilder *pbstrSiteCode);</code>

Parameter

Site_code

Site_code must be a character buffer of at least 30 bytes in size. The function stores the site code in the buffer.

Return Value

Table 23: GetSiteCode() Values

Name	Value	Message	Solution
GSC_OK	0	Function was performed successfully.	
GSC_CRYPTKEY_NOT_INITIALIZED	-1	CrypKey has not been initialized.	This occurs if there is a problem in the InitCrypkey() function. Please refer to the description about this function for more information. If you continue to receive this message, please contact support@crypkey.com .
GSC_ENTRY_FILE_OPEN_FAIL	-2	The entry file could not be opened.	This occurs when you cannot access the .ent file. You must be able to read from and write to this file. If you continue to receive this message, please contact support@crypkey.com .

12.3.25 GetSiteCode2()

GetSiteCode2() returns a pointer to the site code for the program location. The site code must be reported by the user before you can issue a site key.

Usage

If the program detects it is not authorized via the **GetAuthorization()** function, it must, at minimum, show the program's site code and allow the user to enter a site key.

Prototypes

Language	Syntax
C	<code>char *GetSiteCode2 ();</code>
C++	<code>BSTR *GetSiteCode2 ();</code>
VB	<code>GetSiteCode2 () as ByRef String</code>
C#.NET	<code>String *GetSiteCode2 ();</code>

Return Value

This function returns a pointer to the site code.

Note

This function is an optional alternative to using the **GetSiteCode()** function.

12.3.26 InitCrypkey()

InitCrypkey() initializes CrypKey with runtime information; this function must be called before any other CrypKey function.

Usage

Call **InitCrypkey()** at the program startup. For example, if you are using C++, a good place to do this is in the application object constructor.

Prototypes

Language	Syntax
C	<code>int InitCrypkey (char *filepath, char *masterkey, char *userkey, int allow_floppy, unsigned network_max_checktime);</code>
C++	<code>long InitCrypKey (BSTR bstrFilePath, BSTR bstrMasterKey, BSTR bstrUserKey, long nAllowFloppy, long nNetworkTimeout);</code>
VB	<code>InitCrypKey (strFilePath as String, strMasterKey as String, strUserKey as String, nAllowFloppy as Long, nNetworkTimeout as Long) as Long</code>
C#.NET	<code>CKInitErrorEnum InitCrypKey (String *pstrFilePath, String *pstrMasterKey, String *pstrUserKey, int nAllowFloppy, int nNetworkTimeout);</code>

Parameters

Filepath

Filepath is the path of the program file that is checked for authorization. This file name can be the same name as your executable or DLL, or you can choose to use a dummy file if you have more than one application that you would like to protect within your product; simply choose a file name that is always present in your application directory. This dummy file need not contain any information, it simply must reside in the application directory and be specified in the **InitCrypkey()** function call.

Masterkey

Masterkey is the hexadecimal string assigned to your program by CrypKey (Canada) Inc. at purchase time.

Userkey

Userkey is the hexadecimal string assigned to your password by CrypKey (Canada) Inc. at purchase time.

Allow_floppy

Set *allow_floppy* to 1 to allow authorization of a program on a floppy. This is usually set to 0, as the entire floppy can be successfully bit copied.

Network_max_checktime

Network_max_checktime is the time, in seconds, that CrypKey waits for a program to check for authorization under a floating license. After this time, the user is deleted from the queue of users currently using the program or waiting for access to it. This is how CrypKey handles users who may not have logged off properly by using **EndCrypkey()** (perhaps because their computer crashed). If the user attempts to access the program again, the user is put at the end of the queue, behind any other users already using the program or waiting for access. The number you choose for this parameter depends on how often you call **GetAuthorization()**, but it also determines how fast a hung computer is detected by CrypKey and the license on that computer license released for others to use. We recommend that you call **GetAuthorization()** every 5 to 30 minutes, depending on the application. The setting for *network_max_checktime* should be two to three times the **GetAuthorization()** interval.

Return Value

Table 24: InitCrypkey() Values

Name	Value	Meaning	Solution
INIT_OK	0	CrypKey initialization was successful.	
INIT_FILE_NOT_FOUND	-1	The program file could not be located.	This error occurs if the license file that you specified is not located in the application directory. You must ensure that the file that you specified in the <i>filepath</i> parameter of the InitCrypKey() function call is in the location that you stated. If you continue to receive this error, please contact support@crypkey.com .
INIT_MASTERKEY_CRC_FAILURE	-2	The Master Key in the program source code is incorrect.	If you receive this error, ensure the file name in the <i>filepath</i> parameter matches the file name you sent to CrypKey for your Master Key. Also, ensure the Master Key that was sent to you is the same as the Master Key that is in your <i>masterkey</i> parameter in the InitCrypkey() function call. If you continue to receive this error, please contact support@crypkey.com .

Name	Value	Meaning	Solution
INIT_BAD_PRODUCT_NAME	-3	The Master Key does not match the program name (i.e., the Master Key in EXAMPLE.C only works for a program named EXAMPLE.EXE). CrypKey supplies you with a Master Key for your program.	This error occurs if the Master Key in the InitCrypkey() function does not match the file name you specified in the InitCrypkey() function. If you continue to receive this error, please contact support@crypkey.com .
INIT_KEYFILE_CREATION_FAIL	-4	There was no available disk space to write the key file or the directory is write protected.	If you encounter this error, ensure you have full permission to the application directory. CrypKey must be able to read and write directly to the hard drive. If you continue to receive this error, please contact support@crypkey.com .
INIT_NETWORK_NOT_PURCHASED	-5	An attempt is being made to run the program in a network context without purchasing the network license.	If this error occurs, ensure you have the ability to create network licenses. The fastest way to do this is to contact support@crypkey.com .
INIT_NT_NOT_PURCHASED	-6	An attempt is being made to run the program in a Win16 NT context without purchasing the NT license.	If this error occurs, ensure you have the ability to create network licenses. The fastest way to do this is to contact support@crypkey.com .

Name	Value	Meaning	Solution
INIT_NT32BIT_NOT_PURCHASED	-7	An attempt is being made to run the program in a Win32 NT context without purchasing the NT license.	If this error occurs, ensure you have the ability to create network licenses. The fastest way to do this is to contact support@crypkey.com .
INIT_WIN95_NOT_PURCHASED	-8	An attempt is being made to run the program under Windows 95 without purchasing the Windows 95 license.	If this error occurs, ensure you have the ability to create network licenses. The fastest way to do this is to contact support@crypkey.com .
INIT_WIN32S_NOT_PURCHASED	-9	An attempt is being made to run the program under the Win32s context without purchasing the Win32s license.	If this error occurs, ensure you have the ability to create network licenses. The fastest way to do this is to contact support@crypkey.com .
INIT_MULTIPLE_CALL_TO_INIT	-10	This function has already been called during the execution of the protected program.	This error occurs if you call InitCrypkey() more than once before the EndCrypKey() function is called. If you need further assistance, please contact support@crypkey.com .
INIT_THUNK_LIB_NOT_FOUND	-11	The thunk library could not be found.	If this error occurs, ensure you have both crp9516f.dll and cryp95f.dll in the application directory. These files are needed if you are running on Windows 95/98 and your application is 32 bit. If you need further assistance, please contact support@crypkey.com .

Name	Value	Meaning	Solution
INIT_THUNK32_DLL_CORRUPT	-12	A file is corrupt or has been tampered with.	If you need further assistance, please contact support@crypkey.com .
INIT_THUNK16_DLL_CORRUPT	-13	A file is corrupt or has been tampered with.	If you need further assistance, please contact support@crypkey.com .
INIT_32_DLL_CORRUPT	-14	A file is corrupt or has been tampered with.	If you need further assistance, please contact support@crypkey.com .
INIT_THUNK32_DLL_VERSION	-15	A file is outdated.	Ensure you have the distributor files that match the version of CrypKey you are using.
INIT_THUNK16_DLL_VERSION	-16	A file does not match the installed version of CrypKey.	Ensure you have the distributor files that match the version of CrypKey you are using.
INIT_THUNK32_DLL_TAMPERED	-17	A file is corrupt or has been tampered with.	If you need further assistance, please contact support@crypkey.com .
INIT_THUNK16_DLL_TAMPERED	-18	A file is corrupt or has been tampered with.	If you need further assistance, please contact support@crypkey.com .
INIT_THUNK32_DLL_UNKNOWN ERROR	-19	The file has detected an unknown problem.	If you need further assistance, please contact support@crypkey.com .
INIT_THUNK16_DLL_UNKNOWN ERROR	-20	The file has detected an unknown problem.	If you need further assistance, please contact support@crypkey.com .
INIT_PRE_V6_MASTERKEY	-21	You are using a master key from a version previous to version 6.0 of CrypKey.	Contact CrypKey for a version 6.0 master key.

General error values also apply—see Section 12.2: General Error Values.

Note

The *filepath* parameter cannot be just the file name of your program; it will not work under some circumstances under Windows 95. You must carefully provide this function the full path so that it can find your license.

12.3.27 KillLicense()

The **KillLicense()** function is used to disable an existing license and provides a confirmation code that can be entered into the Site Key Generator in order to display the details of the de-authorization.

Usage

You can use this function when you want to remove an existing license and provide your customer with the means of proving that the license is deactivated via the confirmation code.

Prototypes

Language	Syntax
C	<code>int KillLicense (char *confirmCode);</code>
C++	<code>long KillLicense (BSTR *pbstrConfirmCode);</code>
VB	<code>KillLicense (bstrConfirmCode as ByRef String) as Long</code>
C#.NET	<code>CKKillErrorEnum KillLicense (StringBuilder *pbstrConfirmCode); // unsafe</code> <p style="text-align: center;">- or -</p> <code>String *KillLicense (); // safe</code>

Parameter

ConfirmCode

ConfirmCode is a hexadecimal string that can be entered into the Site Key Generator to display information confirming that the license is deactivated.

Return Value

Table 25: KillLicense() Values

Name	Value	Message	Solution
KL_OK	0	Function was performed successfully.	
KL_CRYPTKEY_NOT_INITIALIZED	-1	CrypKey has not yet been initialized.	If this error occurs, obtain the return code from the InitCrypkey() function. Something has gone wrong in the InitCrypkey() function call. InitCrypkey() must be called before this function.
KL_CRYPTKEY_NOT_AUTHORIZED	-2	The protected program is not authorized	This error indicates that the application for which you are trying to kill the license does not have authorization and, therefore, you cannot kill the license.
KL_LICENSE_WRITE_PROTECTED	-3	The license files are currently write protected.	If this error occurs, ensure you are able to read and write directly to the hard drive. If you cannot do this, you cannot perform this function.

Notes

The confirmation code provided by this function can be entered in the Site Code field of the Site Key Generator. Information about the license that was terminated is displayed when the “check” button is pressed.

This function does not remove the special hard-drive signature of the automatic licensing functions.

As commented in the C#.NET prototype, the first code given:

```
CKKillErrorEnum KillLicense (StringBuilder *pbstrConfirmCode);
```

is unsafe code.

The second C#.NET code given:

```
String *KillLicense ();
```

is safe code.

12.3.28 ReadyToTry()

The **ReadyToTry()** function implements a one-time automatic license for a CrypKey-protected application. This license is based on a days time period without version control.

Usage

This function needs to be called only once, just after initialization/authorization. The first time it is called, it takes a minute or so to set up the automatic license. For subsequent calls at the same site, this function takes no action, as it detects that it has already been run.

Prototypes

Language	Syntax
C	<code>int ReadyToTry (unsigned long oplevel, int numDays);</code>
C++	<code>long ReadyToTry (long nOptLevel, long nNumDays);</code>
VB	<code>ReadyToTry (nOptLevel as Long, nNumDays as Long) as Long</code>
C#.NET	<code>CKTryErrorEnum ReadyToTry (int nOptLevel, int nDays);</code>

Parameters

Oplevel

The *oplevel* parameter is the 32 bits that comprise the level and options. The value that is passed here is returned as the *oplevel* parameter in the **GetAuthorization()** function.

NumDays

NumDays is the number of days for which you want the program to be temporarily authorized.

Return Value

Table 26: ReadyToTry() Values

Name	Value	Message	Solution
RTT_OK	0	The function completed successfully.	
RTT_COULD_NOT_GET_SITE_CODE	-1	The function could not get the site code information.	This error occurs if you are not able to retrieve the site code information. You need the ability to read and write directly to the hard drive. If you do not have this ability, you cannot perform this function. If you continue to receive this error, please contact support@crypkey.com .
RTT_COULD_NOT_GET_SITE_KEY	-2	The function could not get the Site Key information.	This error occurs if you are not able to retrieve the site key information. You need the ability to read and write directly to the hard drive. If you do not have this ability, you cannot perform this function. If you continue to receive this error, please contact support@crypkey.com .
RTT_COULD_NOT_SAVE_SITE_KEY	-3	The function could not save the Site Key to the hard drive.	This error occurs if you are not able to retrieve the site code information. You need the ability to read and write directly to the hard drive. If you do not have this ability, you cannot perform this function. If you continue to receive this error, please contact support@crypkey.com .

Name	Value	Message	Solution
RTT_RESERVED_4	-4	Reserved.	
RTT_BAD_DOS	-5	MS-DOS 3.31 or higher must be running on the site.	This error occurs if you or your user is not running a system with MS-DOS 3.31 or higher. The user must use this as the absolute minimum.
RTT_BAD_TRUENAME	-6	Could not get the full pathname of the protected program.	If you receive this error, you ensure the full path name of the CrypKey license file is located in the InitCrypkey() function. If you continue to receive this error, please contact support@crypkey.com .
RTT_NO_REDIRECT	-7	This function does not work on network redirector drives.	This error occurs if you are not running the application off a local drive of the server. If you continue to receive this error, please contact support@crypkey.com .
RTT_NO_DPB	-8	Could not get the drive parameter block from the site hard drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_NO_MEM	-9	Not enough memory was available to execute this function.	This error occurs if there is not enough memory available to perform this function. If you have plenty of memory available, please contact support@crypkey.com .
RTT_CANT_GET_CLUSTER	-10	Could not read a cluster from the site hard-drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .

Name	Value	Message	Solution
RTT_BAD_STAT	-11	Could not get file statistics from the site hard drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_NO_ROOM	-12	The site hard drive has run out of space.	This error occurs if there is not enough memory available to perform this function. If you have plenty of memory available, please contact support@crypkey.com .
RTT_BAD_SECTOR_READ	-13	There is a bad sector read from the site hard drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_BAD_SECTOR_WRITE	-14	There is a bad sector write to the site hard drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_FILE_SEARCH	-15	The file search of the site hard drive has failed.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_FILE_ACCESS	-16	The function could not access the protected program file.	If this error occurs, ensure the program file is in the application directory. If you continue to receive this error, please contact support@crypkey.com .
RTT_FILE_NOT_FOUND	-17	The function could not find the protected program file.	If this error occurs, ensure the program file is in the application directory. If you continue to receive this error, please contact support@crypkey.com .

Name	Value	Message	Solution
RTT_FILE_OPEN	-18	The function could not open the protected program file.	This error occurs when the protected application cannot be opened. Ensure you have full read and write privileges. If you continue to receive this error, please contact support@crypkey.com .
RTT_DONE_THIS	-19	This program has already had its trial period at this hard drive site.	This error occurs if the user tries to execute the ReadyToTry() period more than once on a particular hard drive. This function cannot be run twice. The user must be authorized to avoid this error.
RTT_NO_SIG	-20	The function could not find a signature on the site hard drive.	This error occurs if no signature is found on the hard drive. A signature should be written to the hard drive the first time the software runs. If you continue to receive this error, please contact support@crypkey.com .
RTT_NO_LISTFILE	-21	The function could not find a listfile on the site hard drive.	If this error occurs, ensure you have the most recent CrypKey files. Send information about the files you are using to support@crypkey.com , and we will advise you how to proceed.
RTT_CANT_FIND_DRIVE	-22	The function could not find the site hard drive.	If you receive this error, ensure you have full read and write access to the hard drive. If you continue to receive this error, please contact support@crypkey.com .

Name	Value	Message	Solution
RTT_NO_GOOD_PUTS	-23	The function could not find a good signature on the site hard drive.	If this error occurs, ensure you have the most recent CrypKey files. Send information about the files you are using to support@crypkey.com .
RTT_NO_REAL_DRIVE	-24	The function could not find a real drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_32BIT_FILE_ACCESS	-25	Windows 3.11 32-bit file access must be disabled.	If this error occurs, ensure the 32-bit file access is disabled. If it is not disabled, you cannot run a CrypKey-protected application. If you continue to receive this error, please contact support@crypkey.com .
RTT_CLOSE_ALL_FILES	-26	Another program has a disk lock on the site hard drive.	If you receive this error, please contact CrypKey support at support@crypkey.com .
RTT_MAX_RTTREQ_EXCEEDED	-27	The maximum number of days/runs has exceeded.	This error occurs when the number of days/runs of the trial license has exceeded. You can avoid this error by authorizing the user via the Site Key Generator.

Notes

The **ReadyToTry()** function does not feature version control like its derivative variants **ReadyToTryDays()** and **ReadyToTryRuns()**. For programming purposes, its version number parameter is always considered to be zero.

If the ready-to-try feature is used, it is a good idea to make sure that the user has the ability to access the site code from the application no matter what error code is returned. Then, if the ready-to-try period fails, you can always reauthorize your users for “x” number of runs or days by creating a site key for them.

12.3.29 ReadyToTryDays()

The **ReadyToTryDays()** function implements a one-time automatic license for a CrypKey-protected application. This license is based on a days time period with version control.

This function needs to be called only once, just after initialization/authorization. The first time it is called, it takes a minute or so to set up the automatic license. For subsequent calls at the same site, this function takes no action unless the version number is incremented.

Prototypes

Language	Syntax
C	<code>int ReadyToTryDays (unsigned long oplevel, int numDays, int version, int copies);</code>
C++	<code>long ReadyToTryDays (long nOptLevel, long nNumDays, long nVersion, long nCopies);</code>
VB	<code>ReadyToTryDays (nOptLevel as Long, nNumDays as Long, nVersion as Long, nCopies as Long) as Long</code>
C#.NET	<code>CKTryErrorEnum ReadyToTryDays (int nOptLevel, int nDays, int nVersion, int nCopies);</code>

Parameters

Oplevel

The *oplevel* parameter is the 32 bits that comprise the level and options. The value that is passed here is returned as the *oplevel* parameter in the **GetAuthorization()** function.

NumDays

NumDays is the number of days for which you want the program to be temporarily authorized.

Version

This number is used to distinguish one Automatic License implementation from another, which allows subsequent licenses to be created by newer versions of the protected program. It can be from 1 up to 32.

Copies

This is the number of fixed license copies that you wish to grant the user when this function is run.

Return Value

See Section 12.3.28: `ReadyToTry()` for a list of return values.

Notes

The version parameter normally is begun at 1 and proceeds incrementally upwards from there for each new release of the program.

In order to create a floating license with a specific number of multi-users, you can enter a negative value in the copies parameter. This results in a floating license with a number of users equal to the absolute value of the number entered (e.g., -7 would create a floating license with 7 users).

12.3.30 ReadyToTryRuns()

The **ReadyToTryRuns()** function implements a one-time Automatic License for a CrypKey protected application. This license is based on a runs restriction with version control.

Usage

This function needs to be called only once, just after initialization/authorization. The first time it is called, it takes a minute or so to set up the ready-to-try license. For subsequent calls at the same site, this function takes no action unless the version number is incremented.

Prototypes

Language	Syntax
C	<code>int ReadyToTryRuns (unsigned long oplevel, int numRuns, int version, int copies);</code>
C++	<code>long ReadyToTryRuns (long nOptLevel, long nNumDays, long nVersion, long nCopies);</code>
VB	<code>ReadyToTryRuns (nOptLevel as Long, nNumDays as Long, nVersion as Long, nCopies as Long) as Long</code>
C#.NET	<code>CKTryErrorEnum ReadyToTryRuns (int nOptLevel, int nRuns, int nVersion, int nCopies);</code>

Parameters

Oplevel

The *oplevel* parameter is the 32 bits that comprise the level and options. The value that is passed here is returned as the *oplevel* in the **GetAuthorization()** function.

NumRuns

NumRuns is the number of runs you want the program to be authorized for temporarily.

Version

The *version* number is used to distinguish one automatic license implementation from another, allowing subsequent licenses to be created by newer versions of the protected program. The *version* number can be a value from 1 to 32.

Copies

The *copies* parameter is the number of fixed license copies that you want to grant the user when the function is run.

Return Value

See the **ReadyToTry()** function description in Section 12.3.28: ReadyToTry() for a list of return values.

Notes

The version parameter normally begins at 1 and proceeds incrementally upwards for each new release of the program.

In order to create a floating license with a specific number of multi-users, you can enter a negative value in the copies parameter. This results in the creation of a floating license with a number of users equal to the absolute value of the number entered (e.g., -7 creates a floating license with 7 users).

12.3.31 RegisterTransfer()

The **RegisterTransfer()** function is called from an application that does not yet have authorization and registers the application by placing registration files on a disk or directory. These registration files are used by an application that does have authorization to transfer the authorization.

Usage

The directory for transfer is most likely A:\ because the transfer is usually from one computer to another using a floppy disk. However, this transfer could be from one directory to another on the same computer, or even from one network drive to another. For this reason, the user must be able to enter this parameter.

Prototypes

Language	Syntax
C	<code>int RegisterTransfer (char *directory);</code>
C++	<code>long RegisterTransfer (BSTR bstrDirectory);</code>
VB	<code>RegisterTransfer (bstrDirectory as String) as Long</code>
C#.NET	<code>CKRegisterErrorEnum RegisterTransfer (String *pstrDirectory);</code>

Parameter

Directory

The *directory* parameter is a pointer to a null-terminated text string that contains the path of the directory where some encrypted text files (the registration) are placed.

Return Value

Table 27: RegisterTransfer() Values

Name	Value	Message	Solution
REG_OK	0	The function completed successfully.	
REG_THIS_ALREADY_AUTHORIZED	-1	Only unauthorized sites can receive transferred licenses.	This error occurs if there is already a license present for the application. You can only transfer a license into an unauthorized application. This includes trial licenses.
REG_COULDNOT_OPEN_TARGET_REGFILE	-2	The file could not be opened in the supplied directory due to invalid directory, write protection, disk not ready or no disk space.	If this error occurs, please ensure you have read and write access to the application directory. If you continue to receive this error, please contact support@crypkey.com .
REG_TARGET_ALREADY_REGISTERED	-3	There are already imprint files present in the supplied directory.	This error occurs if the application you try to transfer already has registration. This also occurs if you have tried to register the transfer more than once.
REG_SOURCE_ALREADY_REGISTERED	-4	The site can only have one outstanding registration at a time.	This error occurs if the application you try to transfer already has registration. This also occurs if you have tried to register the transfer more than once.

Name	Value	Message	Solution
REG_CANNOT_OPEN_REGFILE	-5	The matching registration file in the same directory as the program could not be opened.	If this error occurs, please ensure you have read and write access to the application directory. If you continue to receive this error, please contact support@crypkey.com .
REG_CANNOT_WRITE_REGFILE	-6	The matching registration file in the same directory as the application could not be written to.	If this error occurs, please ensure you have read and write access to the application directory. If you continue to receive this error, please contact support@crypkey.com .

General error values also apply — see Section 12.2: General Error Values.

Note

See Section 9: Moving Protected Programs for more information.

12.3.32 SaveSiteKey()

The **SaveSiteKey()** function is used to save to a file the site key issued to the user. The key is checked first before it is saved.

Usage

If the program detects it is not authorized via the **GetAuthorization()** function, it must, at minimum, show the program's site code and allow the user to enter a site key.

Prototypes

Language	Syntax
C	<code>int SaveSiteKey (char *site_key);</code>
C++	<code>long SaveSiteKey (BSTR bstrSiteKey);</code>
VB	<code>SaveSiteKey (bstrSiteKey as String) as Long</code>
C#.NET	<code>CKKeyErrorEnum SaveSiteKey (String *pstrSiteKey);</code>

Parameters

Site_key

The *site_key* parameter is a pointer to the string containing the user-entered site key. Spaces are removed from the key by this function before it is processed.

Return Value

Table 28: SaveSiteKey() Values

Name	Value	Message	Solution
SITE_KEY_OK	0	The function completed successfully.	
SITE_KEY_ENTRY_CHECK_FAIL	-1	The user is likely trying to reuse an old key.	This error occurs if the user attempts to re-enter a site key from a previous site code. The user must give you the site code they currently have and must enter the site key you create for that site code.
SITE_KEY_ENTRY_CRC_FAIL	-2	The key has likely been mistyped.	This error occurs only if the site key the user has entered is incorrect. The most common mistake is to enter a "1" rather than an "l".
SITE_KEY_FILEWRITE_FAILURE	-3	The key file could not be written (disk may be full).	This error occurs if there are no read and write privileges to the application directory. If you continue to receive this error, please contact support@crypkey.com .

Name	Value	Message	Solution
SITE_KEY_NO_LICENSE_TO_ADD_TO	-4	No license exists to which this license can be added.	This can occur if the customer's license expires before he has he used his update key. The customer needs a new authorization.
SITE_KEY_WRONG_ADD_ON_TYPE	-5	Cannot add licenses of different types. Days vs Runs.	Ensure that add-ons to the customer's license have the same restriction type that was present in the original license. For example, you can't add days to a runs-restricted license.

General error values also apply—see Section 12.2: General Error Values.

Note

See Section 5.1: Basic Programming Steps for more information.

GetSiteCode() must be called before the **SaveSiteKey()** function is called. We recommend that you always show the site code to the user before or while allowing the user to enter the site key.

You should always call **GetAuthorization()** to find out the change in the state of the license immediately after this function.

12.3.33 SetNetHandle()

The **SetNetHandle()** function sets the network license handle (i.e. passes the handle of a SLAPI network resource to the program).

Usage

If you want several SLAPI-licensed programs sharing a single network seat, you can use **GetNetHandle()** to get the handle and then distribute it using the **SetNetHandle()** function. The **SetNetHandle()** function should always be called after the call to **InitCrypkey()** is made and before the **GetAuthorization()** call is made.

Prototypes

Language	Syntax
C	<code>void SetNetHandle (unsigned short net_handle);</code>
C++	<code>long SetNetHandle (long nNetHandle);</code>
VB	<code>SetNetHandle (nNetHandle as Long) as Long</code>
C#.NET	<code>void SetNetHandle (int nNetHandle);</code>

Parameter

Net_handle

Net_handle is a handle to a network resource obtained by calling **GetNetHandle()**.

Note

Only one of the programs sharing the network seat can call **EndCrypkey()** successfully—preferably, the last program to stop executing calls this function, since this means that the network seat is clearly no longer required.

12.3.34 TransferIn()

Called from an application that does not have authorization, the **TransferIn()** function imports another authorization license by reading the transfer files from a floppy disk or directory created by the **TransferOut()** function. To be a valid transfer, the site calling this function must have begun the process by using the **RegisterTransfer()** function.

Usage

The directory for transfer is most likely A:\ because the transfer is usually from one computer to another using a floppy disk. However, this transfer could be from one directory to another on the same computer or even from one network drive to another. For this reason, the user must be able to enter this parameter.

Prototypes

Language	Syntax
C	<code>int TransferIn (char *directory);</code>
C++	<code>long TransferIn (BSTR bstrDirectory);</code>
VB	<code>TransferIn (bstrDirectory as String) as Long</code>
C#.NET	<code>CKInErrorEnum TransferIn (String *pstrDirectory);</code>

Parameter

Directory

The *directory* parameter is a pointer to a null-terminated text string that contains the path of the directory where the license files are imported.

Return Value

Table 29: TransferIn() Values

Name	Value	Message	Solution
TI_OK	0	The function completed successfully.	
TI_THIS_ALREADY_AUTHORIZED	-1	A transfer in is allowed by an unauthorized program only.	This error occurs if you try to transfer a license into an application that already has authorization. You can only transfer a license into an unauthorized application. A trial license is considered a license, so you must to kill your trial license before you can transfer in a full authorization.
TI_HARDDISK_REGFILE_NOT_FOUND	-2	Registration file in the same directory as the application is missing.	If you receive this error, your system is not able to find the .reg file needed to perform the transfer. You must have completed the RegisterTransfer() and the TransferOut() functions successfully before you can perform the TransferIn() function.
TI_HARDDISK_REGFILE_CRC_FAILURE	-3	Registration file in the same directory as the application has been damaged or tampered with.	If you receive this error, the .reg file is corrupted. The only way to solve this problem is to delete the .reg file you have and try the transfer process again. If the problem persists, you must request authorization.

Name	Value	Message	Solution
TI_REGFILE_NOT_FOUN D	-4	Registration file not found in the given directory.	This error occurs if the .reg file is not found in the directory specified by the transfer function. This file must be in the correct directory to complete the transfer. The RegisterTransfer() and the TransferOut() functions must be completed successfully before you can perform the TransferIn() function.
TI_REGFILE_CRC_FAIL URE	-5	Registration file in the given directory has been damaged or tampered with.	If you receive this error, the .reg file is corrupted. The only way to solve this problem is to delete the .reg file you have and try the transfer process again. If the problem persists, you must request authorization.
TI_HARDDISK_REGFILE _MOVED	-6	Registration file in the same directory as the application has been moved.	This error occurs if the .reg file is not found in the directory specified by the transfer function. This file must be in the correct directory to complete the transfer. The RegisterTransfer() and the TransferOut() functions must be completed successfully before you can perform the TransferIn() function.
TI_REG_FILES_DONT_ MATCH	-7	A copy of the transfer files from a previous license transfer is probably being used.	This error occurs if the registration file in the directory to which you are trying to transfer is different than the registration information on the floppy disk you are using to transfer.

Name	Value	Message	Solution
TI_SOURCE_HAS_NO_LICENSE	-8	The transfer out operation has not been done yet.	In order to complete the TransferIn() function, you must first complete the RegisterTransfer() and the TransferOut() functions.
TI_SITEKEYFILE_NOT_FOUND	-9	The Site Key file was not found in the given directory.	If you receive this error, please contact support@crypkey.com .
TI_DIFFERENT_SITEKEY	-10	The Site Key file or registration file has been damaged or tampered with.	If you receive this error, the .reg file is corrupted. The only way to solve this problem is to delete the .reg file you have and try the transfer process again. If the problem persists, you must request authorization.
TI_COULDNOT_WRITE_SITEKEYFILE	-11	A write error has occurred in the same directory as the application.	If you receive this error, please contact support@crypkey.com .
TI_RSTKEYFILE_NOT_FOUND	-12	The restriction file was not found in the given directory.	This error occurs if the .rst file is not in the directory specified by the transfer function. This file must be in the correct directory to complete the transfer. In order to complete the TransferIn() function, you must first complete the RegisterTransfer() and the TransferOut() functions.

Name	Value	Message	Solution
TI_DIFFERENT_RSTFILE	-13	The restriction file or the registration file has been damaged or tampered with.	If you receive this error, the .rst file is corrupted. The only way to solve this problem is to delete the .rst file you have and try the transfer process again. If the problem persists, you must request authorization.
TI_COULDNOT_WRITE_RSTKEYFILE	-14	A write error has occurred in the same directory as the application.	This error occurs if you do not have the ability to read and write to the directory. If you continue to receive this error, please contact support@crypkey.com .

General error values also apply—see Section 12.2: General Error Values.

Notes

See Section 9: Moving Protected Programs for more information.

12.3.35 TransferOut()

Called from an application that has authorization, the **TransferOut()** function exports the application's authorization license by placing transfer files on a floppy disk or in a directory where a registration file already exists from a call to the **RegisterTransfer()** function. These transfer files must be used by the same application that created the initial registration files to transfer the license in.

Usage

The directory for transfer is most likely A:\ because the transfer is usually from one computer to another using a floppy disk. However, this transfer could be from one directory to another on the same computer or even from one network drive to another. For this reason, the user must be able to enter this parameter.

Prototypes

Language	Syntax
C	<code>int TransferOut (char *directory);</code>
C++	<code>long TransferOut (BSTR bstrDirectory);</code>
VB	<code>TransferOut (bstrDirectory as String) as Long</code>
C#.NET	<code>CKOutErrorEnum TransferOut (String *pstrDirectory);</code>

Parameter

Directory

The *directory* parameter is a pointer to a null-terminated text string that contains the path of the directory where the license files are exported.

Return Value

Table 30: TransferOut() Values

Name	Value	Message	Solution
TO_OK	0	The function completed successfully.	
TO_THIS_NOT_AUTHORIZED	-1	The application must be authorized in order to do a transfer out.	This error occurs if you try to transfer a license out from an application that does not contain authorization. You can only transfer a license out of an authorized application.
TO_REGFILE_NOT_FOUND	-2	No registration file was found in the given directory.	If you receive this error, your system cannot find the .reg file that is needed to do the transfer. You must complete the RegisterTransfer() function successfully before you able perform the TransferOut() function.
TO_REGFILE_CRC_FAILURE	-3	The registration file has been damaged or tampered with.	If you receive this error, the .reg is corrupted. The only way to solve this problem is to delete the .reg file you have and try the transfer process again. If the problem persists, you must request authorization.

Name	Value	Message	Solution
TO_DIFFERENT_APPLICATION	-4	The transfer is being attempted for a different application.	This error occurs if you try to transfer a license out from a different application than the RegisterTransfer() function has recorded in its license information. You can only transfer a license from the same application into which you are transferring.
TO_TARGET_ALREADY_HAS_LICENSE	-5	The transfer files are already in the given directory.	This error occurs if the TransferOut() function has already been performed once. Please ensure that you only attempt to transfer the license out of the application once.
TO_SITEKEYFILE_NOT_FOUND	-6	The key file in the given directory is missing.	If you receive this message, please contact support@crypkey.com .
TO_COULDNOT_WRITE_SITEKEYFILE	-7	The key file could not be written to the given directory.	This error occurs if you do not have read and write access to the specific directory. If you continue to receive this error, please contact support@crypkey.com .
TO_RSTKEYFILE_NOT_FOUND	-8	The restriction file in the given directory is missing.	If this error occurs, ensure you have a file in your directory with the extension .rst. If the file is not there, try to run the application again. This process should create the file. If this does not work, you must request authorization.

Name	Value	Message	Solution
TO_COULDNOT_WRITE_RST_KEYFILE	-9	The restriction file could not be written to the given directory.	This error occurs if you do not have read and write access to the specific directory. If you continue to receive this error, please contact support@crypkey.com .
TO_COULDNOT_WRITE_REGKEYFILE	-10	The registration file could not be written to the given directory.	This error occurs if you do not have read and write access to the specific directory. If you continue to receive this error, please contact support@crypkey.com .
TO_SOURCE_WRITE_PROTECTED	-11	The source directory is write protected.	This error occurs if you do not have read and write access to the specific directory. If you continue to receive this error, please contact support@crypkey.com .

General error values also apply—see Section 12.2: General Error Values.

Note

See Section 9: Moving Protected Programs for more information.

Appendix I – Sample File

This section contains a file, `example.c`, that shows sample code. To view the `cryptkey.h` file and other files containing 16- and 32-bit sample code, look at the `samples` directory we have provided.

S

everal example files are available in your CrypKey v6.0 installation directories, showing CrypKey function calls from a variety of programming environments. Some samples are supplied by customers. All can be improved upon.

EXAMPLE.C is the best one to look at for a simple demonstration of how function calls work together to protect and license an application.

The 16-bit samples tend to be a bit outdated—check the manual for the latest information if you are using these. The 32-bit samples provide good reference information, as well as some interface ideas.

Each example is a little different, and worth a look just to see how the author uses CrypKey.

The compiled executables for most samples can be found in the DEMOS directory.

Important Note for Visual Basic: because of a property of VB integers, function declarations for 32-bit code and 16-bit code are **not** the same. You can get the function declarations for 32-bit code from the FORJIM sample.

13.1 Example.c

```

/*****
*FILETYPE:- maincode
*FILE      :- EXAMPLE.C
*PURPOSE   :- Simple example of CrypKey Software Library usage
*PROGRAM   :- EXAMPLE.EXE
*LINKS     :- lcrpkey.lib crpkey library large model
*INCLUDES  :- stdio.h crpkey.c stime.h
*COPYRTS   :- CRYPKEY CANADA INC.
*AUTHOR    :- James McCartney
*CREATED   :- June 25, 1992
*****/
#ifdef WIN32
#include <windows.h>
#endif // WIN32
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
#include <dos.h>
#include <malloc.h>
#include "crpkey.h"

/***** Functions in this file *****/
int do_auth(void);
unsigned long time100(); /*gets time of day to 1/100th*/
USHORT challenge(ULONG companyNum, ULONG passNum, USHORT random1, USHORT random2);
int PrintFloatingLicenseSnapshot(void);
long challenge32(long companyNum, long passNum, long random1, long random2);
/***** Functions in this file *****/

/***** Defines and Globals *****/
//These are personalized keys that are issued to you upon purchase of crpkey
#define USER_KEY    "D050 815C D1A2 A79D B103 "

#ifdef WIN32
// Use this if program is 32 bit
#define MASTER_KEY "f2c938d2d34678d2e9217c18d78ea6a8e466cf49520f92cdd1b6916\
bd460d60e7c7b4cc7cc1750bd7188f90ac132b915e82fc8fa60a1d299da0f28ea3c66bd42db\
0be62149daaee4dba55c0e70ce1c13bd343f8b7573abc1e7da0695955ab2bd377f50a9be29a\
04cf816b30cd171e1509ad65100c999e52a35f45a215212a970"
#else
// Use this if program is 16 bit
// #define MASTER_KEY "2A5D 57C4 1B4C 135B F09E 17F7 600B 2D70 79E8 F275 C36A"
#endif

#define CHECKPERIOD 15 //seconds between authorization checks on a network
                        //it is 10 just for demo, 10 min is more practical

char *authTypes[] = {"NONE", "TIME(DAYS)", "RUNS"};
/***** Defines and Globals *****/

/*****
*FUNCTION:- main
*PURPOSE  :- simple demonstration of CrypKey
*ARGUMENT:- n/a
*CALLS    :- The CrypKey API
*RETURNS  :- n/a
*REVISED :- Nov 16, 1997 for Ver 4.3
*****/
main(int argc, char *argv[])
{
//these are used in ckChallenge(), which is a verification function

```

```

#define COMPANYNUM 79560      //these are the correct numbers for example.exe
#define PASSNUM     984534120 //you can use them in your tests if you need to

int done = FALSE;
int ret,err,c,i,num_tries=0;
int authopt,num_allowed,num_used; //restriction info
ULONG level,start_date,start;
char path[400], timestring[30];

char serial[100], model[100], firmware[50];
int GMiret;

printf("CrypKey Test Program path=[%s]\n\n",argv[0]);

/*****
MASTERKEY and USERKEY are issued upon purchase of CrypKey
They insure that Site_Key can only generate keys for your product
*****/
//init crypkey before using any other functions
//printf("Start Init\n");
err = InitCrypkey(argv[0], MASTER_KEY, USER_KEY, FALSE, 3*CHECKPERIOD);
//printf("Done Init\n");

if(err)
{
    printf("Initialization Failure %s(%d) for %s\n",
           ExplainErr(EXP_INIT_ERR,err), err, argv[0]);
    return(0);
}

printf("Initialization OK CrypKey Version %d\n",CrypkeyVersion());

GMiret = GetDrivePermanentSerialData(model, serial, firmware);

if(GMiret)
    printf("GETSERIAL: err=%d %s\n", GMiret, ExplainErr(EXP_HDSN_ERR, GMiret));
else
    printf("\n=====
Drive Data!:
Serial=%s
Model=%s
Firmware=%s\n=====
\n\n", serial,
model, firmware);

printf("Hit a key to continue");
getch();
printf("\n");

err = GetAuthorization(&level, 1); //check authorization, use up 1 run

//Use the challenge function to check the library is not an impostor
//This only needs to be done if you are using the DLL
// It is shown here for demonstration purpose only

if(err==0) //check this only if we think we are authorized
#ifdef WIN32
    //Check new 32 bit challenge
    {
        ULONG random1, random2, result1, result2;
        //generate some random numbers - this can be done any way you like
        random1 = time(NULL);
        random2 = random1*time(NULL);
        result1 = challenge32(COMPANYNUM,PASSNUM/2,random1,random2);
        result2 = CKChallenge32(random1,random2);
        if(result1 != result2)

```



```

        {
            printf("Challenge32 function failed - program aborting\n");
            printf("chal=%u ckChal=%u\n",result1,result2);
            exit(1);
        }
        else
            printf("Challenge32 function successful!\n");
    }
#else
    //use 16 bit challenge function
    {
        USHORT random1, random2, result1, result2; //note USHORT these must be 16
        bit!
        //generate some random numbers - this can be done any way you like
        random1 = (unsigned short)time(NULL);
        random2 = random1*(unsigned short)time(NULL);
        result1 = challenge(COMPANYNUM,PASSNUM,random1,random2);
        result2 = CKChallenge(random1,random2);
        if(result1 != result2)
        {
            printf("Challenge function failed - program aborting\n");
            printf("chal=%u ckChal=%u\n",result1,result2);
            exit(1);
        }
        else
            printf("Challenge function successful!\n");
    }
#endif
else if(err<0) // Only if we are not authorized - Try for a trial license
{
    printf("Attempting to generate a Trial License - this may take a minute\n");
    ret = ReadyToTryDays(33, 3, 1, 1); //Level 33, 3 Days, Version 1, 1 copy
    printf("ReadyToTry returned %d - %s\n",ret, ExplainErr(EXP_RTT_ERR, ret));
}

/*****
Note: this program loops to provide a network example.
Your program only needs to periodically call GetAuthorization if
multiple network copies are to be ran concurrently on a network.
*****/
ungetch(' '); // force it through the loop once to print authorization state
start = 0; //timing loop
while(!done)
{
    if(!kbhit())
    {
        printf("[A=Authorize] [D=Direct Transfer] [L=Acquire License]\n[1=Run
Once] [K=Kill License] [R=Register site]\n[S=Floating License User
Snapshot]\n[O=Tranfer Out] [I=Transfer In] [Q=Quit]\n");
        printf("What's your ambition?: ");
    }

    while(!kbhit())
    {
        if((time(NULL) - start) > CHECKPERIOD)
        {
            num_tries ++;
            start = time100();
            err = GetAuthorization(&level, 0);
            start = time100() - start;
            if (!err)
            {
                printf("\nAuthorization check #%d took %ld ms: %s\n",
num_tries,start*10,
                ExplainErr(EXP_AUTH_ERR,err));
            }
            else

```

```

    {
        printf("\nAuthorization check #d took %ld ms: %s\n",
num_tries,start*10,
            ExplainErr(EXP_AUTH_ERR,err));
    }

    start = time(NULL);
}
}
c = getche();
printf("\n");

switch(c)
{
    case 'A': case 'a':
        do_auth();
        break;

    case 'D': case 'd':
        {
            char dtPath[120];
            printf("Enter the path of the copy of this program to transfer license
to:\n");
            scanf("%s",dtPath);
            err = DirectTransfer(dtPath);
            printf("Direct Transfer: %d %s\n",err,ExplainErr(EXP_DT_ERR,err));
        }
        break;

    case 'L': case 'l':
        {
            char dtPath[120];
            printf("Enter the directory to transfer license from:\n");
            scanf("%s",dtPath);
            err = AcquireLicense(dtPath);
            printf("AcquireLicense: %d %s\n",err,ExplainErr(EXP_AL_ERR,err));
        }
        break;

    case 'Q': case 'q':
        EndCrypkey();
        done=TRUE;
        break;

    case 'R': case 'r':
        printf("Path to place registration file: ");
        scanf("%s",path);
        printf("\n");
        err = RegisterTransfer(path);
        printf("REG: %s\n",ExplainErr(EXP_REG_ERR,err));
        break;

    case 'O': case 'o':
        printf("Path to find registration file: ");
        scanf("%s",path);
        printf("\n");
        err = TransferOut(path);
        printf("TO: %s\n",ExplainErr(EXP_TO_ERR,err));
        break;

    case 'I': case 'i':
        printf("Path to find transfer files: ");
        scanf("%s",path);
        printf("\n");
        err = TransferIn(path);
        printf("TI: %s\n",ExplainErr(EXP_TI_ERR,err));
        break;
}

```

```

    case 'K': case 'k':
    {
        int ret;
        int c;
        char s[50];

        printf("This will delete your license - press Y to verify: ");
        c=getche();
        printf("\n");
        if((c!='Y')&&(c!='y'))
            break;
        ret = KillLicense(s);
        printf("KillLicense() returned %d, Verification: %s\n", ret, s);
        err = GetAuthorization(&level, 0); //update our status
    }
    break;

    case 'l':
        err = GetAuthorization(&level, 1); //check authorization, use up 1 run
        printf("Run Once %s\n", ExplainErr(EXP_AUTH_ERR, err));
        break;

    case 'S': case 's':
        err = PrintFloatingLicenseSnapshot();
        printf("Snapshot returned a %d\nHit any key to continue\n", err);
        start = time(NULL);
        while(!kbhit())
            if((time(NULL) - start) > CHECKPERIOD/2) //don't wait so
long we lose the license
                break;
        while(kbhit()) getch();
        printf("\n");
        break;

    default: break;
}

err = GetAuthorization(&level, 0); // recheck the authorization

printf("=====\n");
printf("This file is %s authorized at this site: %s\nLEVEL=%ld\n",
    (err==0)? "now": "NOT", ExplainErr(EXP_AUTH_ERR, err), level&0x00FF);

printf("OPTIONS Available: ");
//Note that this program has 8 options defined in skw.ini
//Therefore, we check the top 8 bits of 'level' for these options
//We could also use get_option, a new function that is easier to use
for(i=31; i>23; i--)
    if((1L<i)&level)
        printf("#%d ", 32-i);

// display restrictions
ret = GetRestrictionInfo(&authopt, &start_date, &num_allowed, &num_used);
if(ret==0)
    printf("RESTRICTIONS: TYPE:%s START:%s \n#ALLOWED:%d #USED:%d\n",
        (authopt<0)? "N/A": authTypes[authopt], CKTimeString(timestring, start_date),
        num_allowed, num_used);
    printf("Number of Copies allowed from this site: %d\n", get_num_copies());
    printf("Number of Network Users allowed from this site:
%d\n", get_num_multi_users());

    /*
    {
        int serr, stz;
        serr = GetServerTimeZone(&stz);

```

```

        printf("Server Time Zone=%d err=%d\n",stz,serr);
    }

    */

printf("=====\n\n");

#ifdef TEST_OTHER_CALLS //define this to see other calls in action
printf("Sitecode is %s\n",GetSiteCode2()); //some languages need to use this
//intead of get_site_code()

printf("Level = %d\n", get_level(8));
for(i=1;i<=8;i++)
    printf("Opt%d=%d ", i,get_option(8,i));
printf("\n");

printf("RESTRICTIONS: TYPE:%s #ALLOWED:%d #USED:%d\n",
    authTypes[Get1RestInfo(1)],Get1RestInfo(2),Get1RestInfo(3));
ExplainErr2(1, -2, text);
printf("Error text for func 1 err2 is\n  [%s]\n",text);

printf("=====\n\n");
#endif
}

EndCrypkey();
return(0);
}

/*****
*FUNCTION: do_auth()
*PURPOSE : does the SiteCode - SiteKey() transaction with the customer
*ARGUMENT: none
*RETURNS : non zero if error
*****/
int do_auth()
{
    int err;
    char site_code[50],key[50];

    if(err = GetSiteCode(site_code)) //let the user authorize
    {
        printf("GET SITE CODE:%s %d\n", ExplainErr(EXP_GET_SITECODE_ERR,err),err);
        //printf("Initialization Failure - '%s' is not the path of this file\n");
        return(0);
    }

    printf("Site Code: %s\nEnter Site Key or '.' to quit: ",spc(site_code));
    key[0]=0;
    while(!kbhit()); // wait for entry
    gets(key);

    if(key[0]=='.')
        return(0);

    err = SaveSiteKey(key);

    printf("%s\n", ExplainErr(EXP_SAVE_SITEKEY_ERR,err));

    return(0);
}

/*****
*FUNCTION: time100()
*PURPOSE : gets time of day in 100s of seconds

```

```

*ARGUMENT: n/a
*RETURNS : time of day in hundreds of seconds
*****/
unsigned long time100() /*gets time of day to 1/100th*/
{
#ifdef WIN32
    SYSTEMTIME t;
    GetSystemTime(&t);
    return ((t.wHour*60 + t.wMinute)*60 + t.wSecond)*100 + t.wMilliseconds/10;
#else
    struct dostime_t t;
    _dos_gettime(&t);
    return ((t.hour*60 + t.minute)*60 + t.second)*100 + t.hsecond;
#endif // not WIN32
}

/*****
*FUNCTION: challenge()
*PURPOSE : does the calculation for the CrypKey ckChallenge verification
*ARGUMENT: companyNum - assigned by Kenonic
           passNum     - assigned by Kenonic
           random1     - make one up yourself
           random2     - make one up yourself
*RETURNS : non zero if error
*****/
USHORT challenge(ULONG companyNum, ULONG passNum, USHORT random1, USHORT
random2)
{
    int i;
    ULONG ret, r1, r2;

    r1 = (ULONG)random1;
    r2 = (ULONG)random2;

    ret = 0;

    for(i=0;i<11;i++)
        ret = (ret * r1 + companyNum)%16381L + (ret * r2 + passNum)%16369L;

    return (USHORT)ret;
}

int PrintFloatingLicenseSnapshot(void)
{
    FLS_REC *buf;
    int ret, numEntries, i;
    unsigned long bufSize;
    char timestring[30];

    //allocate a buffer
    bufSize = (unsigned long)sizeof(FLS_REC) * ABS_MAX_USERS;
    if( (buf = (FLS_REC *)malloc(bufSize)) == NULL )
    {
        printf("Could not allocate enough space for this!\n");
        return -99;
    }
    ret = FloatingLicenseSnapshot(bufSize, &numEntries, buf);

    if(ret!=0)
        printf("Error %d - %s\n",ret,ExplainErr(EXP_FLS_ERR, ret));
    else
    {
        printf("\nFloating License Snapshot: %d Entries ret=%d\n\n",numEntries,ret);
        printf("Status Date/Time Started      ComputerName      UserName\n");
        printf("=====\n");
    }
}

```

```

        for(i=0;i<numEntries;i++)
            printf("%-6d %s %15s %15s\n", buf[i].status,
                CKTimeString(timestring,buf[i].starttime),
                buf[i].computerName, buf[i].userName);
        printf("=====\n");
    }

    free(buf);
    return(ret);
}

#ifdef WIN32
/*****
*FUNCTION: challenge32()
*PURPOSE : does the calculation for the CrypKey ckChallenge32 verification
*ARGUMENT: companyNum - assigned by Kenonic
            passNumDiv2- assigned by Kenonic
            random1    - make one up yourself
            random2    - make one up yourself
*RETURNS : non zero if error
*NOTES:    This function uses longs, not unsigned longs, to be VB friendly
            If unsigned longs are accidentally used, it will not work
*****/
long challenge32(long companyNum, long passNumDiv2, long random1, long
random2)
//VB users should divide their PassNum by 2 manually,before coding it,
//to get PassNumDiv2
{
    int i;
    long ret;

    ret = 1;
    for(i=2;i<11;i++)
    {
        ret = ret%32769 * ( (random1/i)%32769 + (companyNum/i)%32769);
        ret = ret%32769 * ( (random2/i)%32769 + (passNumDiv2/i)%32769);
    }

    return(ret);
}
#endif

```

Appendix II – Quick Reference

14.1 CrypKey Functions

Following are brief descriptions of CrypKey functions and their C prototypes. For more details, including their prototypes, see Chapter 12: Function Reference.

AcquireLicense()

allows an unlicensed CrypKey protected program to acquire a license from a licensed copy.

```
int AcquireLicense (char *licensePath);
```

CKChallenge()

Verifies that the CryKey DLL has not been replaced with an impostor DLL.

```
unsigned int CKChallenge (unsigned int random1, unsigned int random2);
```

CKChallenge32()

Verifies that the CryKey DLL has not been replaced with an impostor DLL.

```
long CKChallenge32 (long random1, long random2);
```

CKTimeString()

Translates, into a text string, the ULONG time values returned by the functions GetAuthorization() and FloatingLicenseSnapshot().

```
char * FUNCTYPE CKTimeString(char *timestringbuf, ULONG t)
```

CrypKeyVersion()

returns the version number of the CrypKey system currently in use.

```
int CrypKeyVersion ();
```

DirectTransfer()

Called from an application that has authorization, this function transfers the application's authorization license to another directory.

```
int DirectTransfer(char far *directory);
```

EndCrypkey()

Notifies CrypKey that the program is stopping. This function is only necessary if you are using Network features.

```
void EndCrypkey();
```

ExplainErr()

Returns a text string explaining the return code of various functions.

```
char far *ExplainErr(int functioncode, int errcode);
```

ExplainErr2()

Returns a text string explaining the return code of various functions. This function is similar to the Explain Err() function, but your error message is written into your string text.

```
void ExplainErr2(int functioncode, int errcode, char *text);
```

FloatingLicenseGetRecord()

Used only in COM Objects and C#.NET Assemblies: called from an application that has authorization, this function returns a single record of the current floating license status.

[Prototypes available for C++ and C#.NET but not for C or VB.]

FloatingLicenseSnapshot()

Returns a snapshot of all users holding or waiting for a network floating license.

```
int FloatingLicenseSnapshot (unsigned long bufSize, int *numEntries, FLS_REC *buf);
```

FloatingLicenseTakeSnapshot()

Used only in COM Objects and C#.NET Assemblies: returns a snapshot of all users holding or waiting for a network floating license. [Prototypes available for C++ and C#.NET but not for C or VB — see Sec. 12.3.11: FloatingLicenseSnapshot.]

Get1RestInfo()

Used to return one of three different types of data depending on the value of the “which” input parameter.
 int **Get1RestInfo**(int which);
 “which” can be 1 — returns authopt, 2 — returns num_allowed, 3 — runs num_used

GetAuthorization()

Gets the level at which the program is authorized to run or an authorization failure code. Also used to decrement the number of uses count. Must be called, together with InitCrypKey, before any other CrypKey function.
 int **GetAuthorization**(unsigned long *oplevel, int decrement);

GetAuthorization2()

Determines if the program is authorized to run, and decrements the license count — but does not obtain option and level information.
 int **GetAuthorization2** (int decrement);

GetDrivePermanentSerialData()

proprietary CrypKey software function that accesses the following hard drive information on a customer's computer: model number, serial number, firmware data.

GetLastError()

Returns the text for the last error reported.
 [Prototypes available for C++, VB and C#.NET but not for C.]

GetLastErrorCode()

C#.NET function only: returns the number of the last error reported.
 long **GetLastErrorCode**();

GetLevel()

Used to determine the level at which your program has been authorized to run.
 Unsigned long **GetLevel**(int numDefineOpts);

GetNetHandle()

Gets and returns the network license handle of the protected program.

GetNumCopies()

Gets the number of license copies that the current site has been granted.

GetNumMultiUsers()

Gets the number of multi-users this site has been granted.
 int **GetNumMultiUsers**();

GetOption()

Used to determine the status of a specific option.
 int **GetOption**(int numDefineOpts, int optnum);

GetRestrictionInfo()

Gets information on any restrictions that are present in the license.
 int **GetRestrictionInfo** (Int far *authopt, ULONG far *start_date, int far *num_allowed, int far *num_used);

GetSiteCode()

Gets the site code for this program location. The site code must be reported by the customer before the developer can issue the site key.
 int **GetSiteCode**(car far *site_code);

GetSiteCode2()

Returns a pointer to the site code for this program location.
 char ***GetSiteCode**();

InitCrypkey()

Initializes CrypKey with runtime information. This function and **GetAuthorization()** must be called before any other function in CrypKey.
 int **InitCrypkey**(char far *filepath, char far *masterkey, char far *userkey, int allow_floppy, unsigned network_max_checktime);

KillLicense()

Disables an existing license and provides a confirmation code that can be entered into the Site Key Generator in order to display the details of the de-authorization.
 int **KillLicense** (char *confirmCode);

ReadyToTry()

Implements a one-time automatic license based on a “days” time period, **without** version control.
 int **ReadyToTry** (unsigned long oplevel, int numDays);

ReadyToTryDays()

Implements a one-time automatic license based on a “days” time period, **with** version control.
 int **ReadyToTryDays** (unsigned long oplevel, int numDays, int version, int copies);

ReadyToTry_Runs()

Implements a one-time automatic license based on a “runs” restriction **with** version control.

int ReadyToTryRuns (unsigned long oplevel, int numRuns, int version, int copies);

RegisterTransfer()

Called from an application that does not yet have authorization, this function registers the application by placing license imprint files on a disk or in a directory.

int **RegisterTransfer**(char far *directory)

SaveSiteKey()

This function is used to save to file the site key that has been acquired from the user. The key is checked before it is saved.

int **SaveSiteKey**(char far *site_key);

SetNetHandle()

Sets the network license handle, i.e. passes the handle of a SLAPI network resource to the program.

void SetNetHandle (unsigned short net_handle);

TransferIn()

Called from an application that does not have authorization, this function transfers the application's authorization license in by reading transfer out license imprint files on a disk or in a directory.

int **TransferIn**(char far *directory);

TransferOut()

Called from an application that has authorization, this function transfers the application's authorization license out by placing license imprint files on a disk or in a directory.

int **TransferOut**(char far *directory);

14.2 CrypKey Return Codes

InitCrypkey()		GetAuthorization()	
Defined Name within CRYPTKEY.H	Return Value	Defined Name within CRYPTKEY.H	Return Value
INIT_OK	0	AUTH_OK	0
INIT_FILE_NOT_FOUND	-1	AUTH_INIT_FAIL	-1
INIT_MASTERKEY_CRC_FAILURE	-2	AUTH_DISALLOW_FLOPPY	-2
INIT_BAD_PRODUCT_NAME	-3	AUTH_BAD_PATH	-3
INIT_KEYFILE_CREATION_FAIL	-4	AUTH_NOT_PRESENT	-4
INIT_NETWORK_NOT_PURCHASED	-5	AUTH_DIFFERENT	-5
INIT_NT_NOT_PURCHASED	-6	AUTH_BAD_MASTERKEY	-6
INIT_NT32BIT_NOT_PURCHASED	-7	AUTH_SITEKEY_CRC	-7
INIT_WIN95_NOT_PURCHASED	-8	AUTH_TIME_TOO_EARLY	-8
INIT_WIN32S_NOT_PURCHASED	-9	AUTH_TIME_SETBACK	-9
INIT_MULTIPLE_CALL_TO_INIT	-10	AUTH_TIME_RUNOUT	-10
INIT_THUNK_LIB_NOT_FOUND	-11	AUTH_RUNS_RESTR	-11
INIT_THUNK32_DLL_CORRUPT	-12	AUTH_NOT_ENOUGH_RUNS	-12
INIT_THUNK16_DLL_CORRUPT	-13	AUTH_MISSING_RST_FILE	-13
INIT_32_DLL_CORRUPT	-14	AUTH_RST_BAD_CRC	-14
INIT_THUNK32_DLL_VERSION	-15	AUTH_RST_BAD_LOCATION	-15
INIT_THUNK16_DLL_VERSION	-16	AUTH_ENTRY_CHECK_FAIL	-16
INIT_THUNK32_DLL_TAMPERED	-17	AUTH_NETTABLEFILE_FAIL	-17
INIT_THUNK16_DLL_TAMPERED	-18	AUTH_NETMAX_EXCEEDED	-18
INIT_THUNK32_DLL_UNKNOWN_ERROR	-19	AUTH_NETWORK_NOT_ALLOWED	-19
INIT_THUNK16_DLL_UNKNOWN_ERROR	-20	AUTH_RSTFILE_WRITE_PROTECT	-20
INIT_PRE_V6_MASTERKEY	-21	AUTH_TIME_CLOCK_TAMPERING	-21
		KEY_BAD_LOCATION	-22

14.3 CrypKey 6.X OS File Distribution Matrix

FILENAME	WIN9X	WIN ME	WIN NT	WIN 2000	WIN XP	NOVELL
crp95f.dll ⁵	✓	✓			▪	▪
crp9516f.dll ⁵	✓	✓			▪	▪
ck16rmv.exe ⁵	✓	✓			▪	▪
lcrypkyd.dll	❖	❖	❖	❖	❖	❖
crp32dll.dll	❖	❖	❖	✓	✓	✓
hdsnl.dll	✓ ❖	✓ ❖	✓ ❖	✓ ❖	✓ ❖	✓ ❖
crp32001.ngn	✓	✓	✓	✓	✓	✓
wckserve.exe ²	✓	✓				
setupex.exe ¹			✓	✓	✓	
cks.exe ¹			✓	✓	✓	
ckserver.exe	✓	✓	✓	✓	✓	✓
ckserver.nlm						✓
cknetwk.exe	❖	❖	❖	❖	❖	❖

Legend:

- ❖ Indicates files required for 16Bit applications.
- ✓ Indicates files required for 32Bit applications.
- Indicates optional files depending on XP Home Edition/Pro compatibility mode settings.

Cells with a gray background indicate files required for a network.

Files in *italics* are specific to **CrypKey Instant**.

Files in gray text are specific to **CrypKey SDK**.

Files in regular black text are common to both SDK and Instant.

CRYPKEY INSTANT – refer to Sections 3.6.1 and 3.6.2 in the CrypKey Instant Manual

CRYPKEY SDK – refer to Sec. 2.6, Libraries and Distribution Files and Table 3: Libraries, in the CrypKey SDK manual

Notes

1. setupex.exe and cks.exe are files used to install the NT driver (Crypkey License Service) required on all NT systems whether networked or stand-alone.
2. wckserve.exe is required for Win9X/ME servers
3. splash.int is required for custom splash screen. See online database for more information.
4. See online database for more information regarding language support.
5. Thunk files for Win9X/ME go into the application or license directory only.
6. If you use static libraries, crp32d11.dll or lcrypkyd.dll do not need to be distributed.

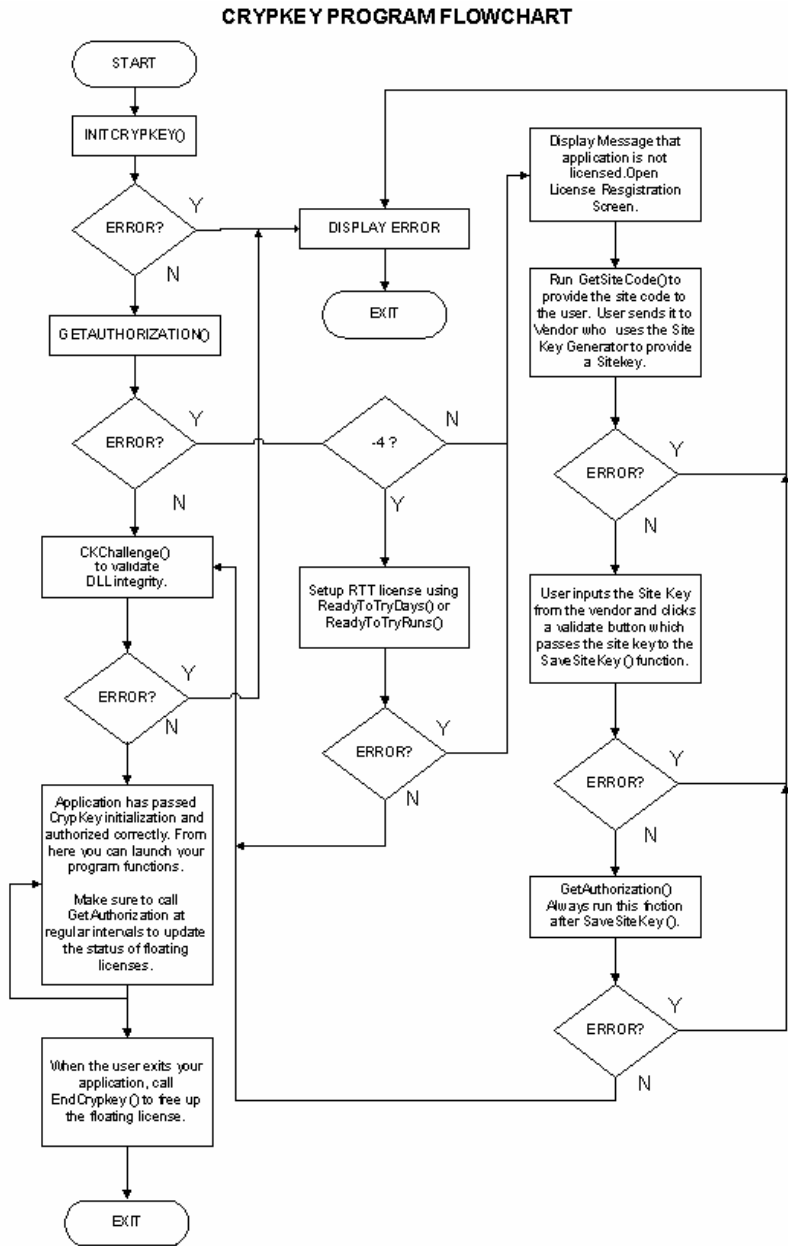


Figure 57: CripKey Program Flowchart

Example: Running CrypKey on a Win9X/ME peer to peer network


Win9X/ME Server
Name = APPSERVER


Win9X/ME Client

This System is acting as the server. Other clients will connect to this computer to run the application:

C:\APP\EXAMPLE.EXE

1. Load the WCKSERVE driver on the system by placing the following lines in the autoexec.bat file:

```
SET WCKSERVE=C:\APP
C:\APP\WCKSERVE.EXE
```

or you can have one line instead of the two above by using the following statement:

```
C:\APP\WCKSERVE.EXE C:\APP
```

Note: The wckserve.exe file must exist in the directory specified.

2. You must share the directory C:\APP so that other users on the network will have FULL access to it.

3. Reboot the computer for the changes to take effect.

This is a client that will run the application from the server. It also requires some configuration:

1. Load the WCKSERVE driver on the system by placing the following lines in the autoexec.bat file:

```
SET WCKSERVE=\\APPSERVER\C\APP
\\APPSERVER\C\APP\WCKSERVE.EXE
```

or you can have one line instead of the two above by using the following statement:

```
\\APPSERVER\C\APP\WCKSERVE.EXE \\APPSERVER\C\APP
```

Note: The wckserve.exe file must exist in the directory specified.

2. Reboot the computer for the changes to take effect.

3. Create a shortcut to the executable on APPSERVER.

Figure 58: Example Win9X/ME CrypKey Configuration

Glossary

This glossary contains definitions of terms used in this manual.

T

he terms commonly used in this manual are defined in this section. These definitions help you understand the CrypKey SDK product and its protection strategies.

.NET

.NET is a framework for objects and how they work together. .NET objects are language-neutral and may be written in C#, C++, Visual Basic, J #, or other languages. The code generated for .NET is an intermediate language that needs the Common Language Runtime (CLR) to interpret the code. The use of the CLR allows .NET code to run on different operating systems and platforms without having to recompile the code.

.NET Assembly

A .NET assembly is a file containing .NET classes. The assembly contains class definitions that allow other .NET programs to identify and use the classes within it.

Automatic License

This is a license that is created by using the **ReadyToTry()** function or one of its variants. Such licenses are created automatically by the software without requiring you or your distributor to provide a site key.

Common Object Model (COM)

The Common Object Model is a specification for objects and how they work together. COM objects are language-independent and may be written in C++, Visual Basic, Java, or other languages. The code generated for COM is standard Windows machine executable binaries.

COM Object

A COM object is a file containing COM classes. The COM object contains class definitions that allow other COM programs to identify and use the classes within it.

Company Number

This is one of two special keys used in the **CKChallenge()** function (the other key is the password number). The company number key is based on a private number that CrypKey assigns to your company.

Developer Keys

These are the four keys that CrypKey issues you after receiving payment. These include:

- company number
- password number
- Master Key
- User Key

Fixed License

A product license usable only on the computer where it was installed (node-locked license). Although the program can be copied to other computers, its inherent security ensures it can only run on the computer where it was installed.

Floating License

A product license that allows a pre-determined number of product copies to run concurrently on various client computers from a single, networked server copy.

Level

A parameter defined for a product license. The Site Key Generator communicates the level to your product via the site key, enabling you to control how your product runs and which instance of your product is authorized. CrypKey SDK communicates only the level value to the protected application; you must implement the action(s) within the product based on the level.

License

A protected product's authorization is defined by the license. The license contains details of the authorization, including levels, options, license type, and license restriction. In physical terms, the license consists of four hidden system files that reside in the same directory location as the protected program.

Master Key

A hexadecimal number issued by CrypKey, based on your company number and product name. To use CrypKey with your product, you need a Master Key and User Key (these keys are entered into the **InitCrypkey()** function).

Option

Selection of a specification that may include or exclude a component as part of a product license. Together, the numeric symbols for the options and levels defined for your product comprise a 32-bit number used as part of the site key.

Password Number

One of two special keys used in the **CKChallenge()** function when using the CrypKey API (the other key is the company number). The password number is based on the product password, which you specify. CrypKey creates the password number from your product password and provides it to you.

Site

In software licensing, the internal boot device (hard drive) of the computer where a protected program is installed. The term is generally synonymous with the computer itself.

Site Code

A site-specific hexadecimal number issued by a protected product when installed on a client's computer. The site code is provided by the **GetSiteCode()** function. Your customers provide the site code when requesting licensing changes for the product.

Site Key

A hexadecimal number based on a client's site code, created by via the Site Key Generator. The site key unlocks client software features and license restrictions.

User Key

A hexadecimal number issued by CrypKey, based on your product password. To use CrypKey with your product, you need a Master Key and a User Key (enter these keys in the **InitCrypkey()** function).

Index

This index includes page references for significant occurrences of topics and terms.

B y “significant occurrences”, we mean appearances of a term where information is provided about its meaning or use in the CrypKey context. Where appropriate, we have also provided “see also” cross-references between terms.

▪

.NET assembly42, 48, 96, 166, 182, 196

.NGN file20, 47, 87, 94, 150, 162, 165, 255

A

AcquireLicense()..... 181

anti-virus software 148

authorization transfer

 into target computer 92

 out of source computer 92

 registering 91

authorizing

distributors.....	80
MS-DOS program	57
Site Key Generators.....	78
Windows program	51
your program.....	91
automatic license.....	259

C

CD-ROM distribution	146
certification.....	148
CKChallenge()	10, 95, 126, 160, 169, 170, 171, 172, 251
CKChallenge32()	172, 173
cks.exe.....	131
ckserve.exe.....	44, 132
ckserver.nlm	45, 131
client annoyance.....	142
clock manipulation	146
CloneBuster	18, 63
COM Object.....	42, 48, 93, 96–106, 182, 184
company number.....	10, 260
computer signatures	143
CRP32001.NGN file	20, 47, 94
cryp9516f.dll	48
cryp95f.dll	48
CrypKey Instant	16
CrypKey SDK	
features	1, 17
installation	24
trial period	2

cryptkey.h	92, 179, 181, 202, 240
CrypkeyVersion()	160, 174
cui.exe	106

D

DAL	78
days license	61
defragmentation programs	144
developer keys	3, 9, 260
obtaining	10
temporary	11
development	
platforms	8
process	15
tools	95
direct transfer	135
DirectTransfer()	135, 160, 175, 177, 180, 181, 251
disk compression	144
Distributor Authorizing License	78
distributors, authorizing	80

E

EndCrypkey()	90, 127, 160, 178, 198, 208, 230, 251
errors	110, 155, 162
codes	127
Novell NetWare	155
example.exe	8, 11, 58
Example.exe	51
ExplainErr()	88, 89, 90, 149, 160, 162, 179, 180, 251

ExplainErr2()	180, 182
exporting to United States	148
external links	
security	107
using binary files	106

F

features	1, 17
file set protection	147
fixed license	260
floating license	
defined	260
specifying	130
FloatingLicenseSnapshot()	181, 183
floppy transfer	135
foreign system compatibility	147
functions	159

G

Get1RestInfo()	160, 185, 186
GetAuthorization()	88, 89, 127, 130, 160, 171, 173, 180, 181, 187, 204, 205, 227, 229
GetAuthorization2()	161, 194, 195
GetDrivePermanentSerialData()	181, 196
GetLevel()	161, 194, 197
GetNetHandle()	161, 198, 229
GetNumCopies()	161, 199
GetNumMultiUsers()	161, 199
GetOption()	161, 194, 200

GetRestrictionInfo()	161, 201, 202
GetSiteCode()	89, 161, 204, 229, 262
GetSiteCode2()	161, 205
guarantee	16

H

hacking	125
hard drives	
failure	144
removable	147

I

information transfer	107, 108
InitCrypkey()	88, 89, 127, 161, 171, 173, 180, 181, 194, 195, 206, 261, 262
installer software	148
installing	
32-bit thunk library	48
CrypKey SDK	24
MS-DOS network driver	44
network driver	130
Novell NetWare	45
Windows 95 network driver	43
Windows NT driver	43
Internet distribution	147

K

keys	3
master	3, 9
obtaining	10
site	3, 10, 11
temporary	11
user	3, 9
KillLicense()	161, 180, 181, 213

L

levels.....59, 60, 194, 261

libraries

 linking93

 third party144

licenses

 changing.....76

 defined17, 261

 establishing88

 floating.....145

 limiting61

 moving.....135

 preventing loss.....157

 transferring.....136, 143

M

Master Key.....3, 9, 261

MS-DOS

 network driver.....44

 program.....57

multiple program names145

N

network licensing130

Norton Utilities Speed Disk.....157

Novell NetWare45

O

options59, 60, 194, 261

P

packing list.....40

password	
number	10, 261
temporary (for development)	11

R

readme	41
readme!.hlp	40
ReadyToTry()	161, 180, 181, 215, 221, 259
ReadyToTryDays()	161, 222
ReadyToTryRuns()	161, 223
RegisterTransfer()	161, 180, 181, 225, 230
releases	145
run modes	
silent	111
verbose	111
runs license	61

S

SaveSiteKey()	89, 162, 180, 181, 187, 227, 229
SetNetHandle()	162, 198, 229
setupex.exe	110, 131
setupex.xco	110
silent mode	111
site 261	
site code	16, 262
obtaining	11
transferring	17
site key	3, 10, 262
creating	11

transferring.....	17
Site Key Generator	
defined	59
software protection	139
Stealth directory.....	42, 43
StealthPlusTM	115–23
StealthPLUS™	18, 19, 42, 43
support.....	8, 143
supported platforms	131
system requirements	7

T

tampering.....	125
telephone authorization	142
testapp.exe	51
thunk library	48
TransferIn()	162, 180, 181, 187, 230
TransferOut()	162, 180, 181, 187, 235
trial period	2

U

unlimited license	61
updates	145
User Key	3, 9, 262

V

verbose mode.....	111
-------------------	-----

W

Watcom..... 144

wckserve.exe43, 131

Windows 95 network driver 43

Windows 98 148

Windows NT

 driver installation program..... 109

 supported versions..... 113

 uninstalling driver 112

Windows NT drivers43

Windows program

 authorizing.....51, 52