> *Aladdin is the #1 vendor in the Software Licensing Authentication Tokens market for 2002 and 2003.*
>
> — *IDC Bulletin #31432, 2004* —

eAladdin.com

HASP® HL

REINVENTING SOFTWARE PROTECTION & LICENSING

Software Protection and Licensing Guide

Aladdin®

SECURING THE GLOBAL VILLAGE

eAladdin.com

COPYRIGHTS AND TRADEMARKS

ALADDIN KNOWLEDGE SYSTEMS LTD.
DEVELOPER'S LICENSE AGREEMENT

IMPORTANT INFORMATION - PLEASE READ THIS AGREEMENT CAREFULLY BEFORE OPENING THE PACKAGE AND/OR USING THE CONTENTS THEREOF AND/OR BEFORE DOWNLOADING OR INSTALLING THE SOFTWARE PROGRAM. ALL ORDERS FOR AND USE OF THE HASP AND/OR HASP HL FAMILY PRODUCTS (including without limitation, the Kit, libraries, utilities, diskettes, CD ROM, HASP® and/or HASP®HL keys, the software component of Aladdin's HASP and/or HASP HL and the HASP HL Protection and Licensing Guide) (hereinafter “Product”) SUPPLIED BY ALADDIN KNOWLEDGE SYSTEMS LTD. (or any of its affiliates - either of them referred to as “ALADDIN”) ARE AND SHALL BE, SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

BY OPENING THE PACKAGE CONTAINING THE PRODUCTS AND/OR BY DOWNLOADING THE SOFTWARE (as defined hereunder) AND/OR BY INSTALLING THE SOFTWARE ON YOUR COMPUTER AND/OR BY USING THE PRODUCT, YOU ARE ACCEPTING THIS AGREEMENT AND AGREEING TO BE BOUND BY ITS TERMS AND CONDITIONS.

IF YOU DO NOT AGREE TO THIS AGREEMENT OR ARE NOT WILLING TO BE BOUND BY IT, DO NOT OPEN THE PACKAGE AND/OR DOWNLOAD AND/OR INSTALL THE SOFTWARE AND PROMPTLY (at least within 7 days from the date you received this package) RETURN THE PRODUCTS TO ALADDIN, ERASE THE SOFTWARE, AND ANY PART THEREOF, FROM YOUR COMPUTER AND DO NOT USE IT IN ANY MANNER WHATSOEVER. UPON RETURNING THE PRODUCT WITH A COPY OF THE SALES RECIEPT TO ALADDIN YOU WILL RECEIVE A REFUND.

1 Title & Ownership

THIS IS A LICENSE AGREEMENT AND NOT AN AGREEMENT FOR SALE. The software component of the Product, including any revisions, corrections, modifications, enhancements, derivative works, updates and/or upgrades thereto, (hereinafter in whole or any part thereof defined as: “Software”), and the related documentation, ARE NOT FOR SALE and are and shall remain in Aladdin's sole property. All intellectual property rights (including, without limitation, copyrights, trade secrets, trademarks, etc.) evidenced by or embodied in and/or attached/ connected/related to the Product, (including, without limitation, the Software code and the work product performed in accordance with Section 2 hereunder) are and shall be owned solely by Aladdin. This Agreement does not convey to you an interest in or to the Software but only a limited right of use revocable in accordance with the terms of this Agreement. Nothing in this Agreement constitutes a waiver of Aladdin's intellectual property rights under any law.

2 License

Subject to payment of applicable license fees, Aladdin hereby grants to you, and you accept, a personal, nonexclusive and fully revocable limited license to use the Software, in executable form only, as described in the Software accompanying user documentation and only according to the terms of this Agreement: (i) you may install the Software and use it on computers located in your place of business, as described in Aladdin's related documentation; (ii) you may merge and link the Software into your computer programs for the sole purpose described in the HASP HL Protection and Licensing Guide; however, any portion of the Software merged into another computer program shall be deemed as derivative work and will continue to be subject to the terms of this Agreement; and (iii) you are permitted to make a reasonable number of copies - but not more than three (3) - of the Software solely for development in accordance with the HASP HL Protection and Licensing Guide and backup purposes. The Software shall not be used for any other purposes.

3 Sub-licensing

After merging the Software in your computer program(s) according to section 2 above, you may sub-license, pursuant to the terms of this Agreement, the merged Software and resell the hardware components of the Product which you purchased from Aladdin, to distributors and/or users. Preceding such a sale and sub-licensing, you shall incorporate by reference in your contracts with such distributors and/or users, and otherwise provide for all distributors and/or users to be bound by, the warranties, disclaimers, and license terms specified by Aladdin in this Agreement.

4 Prohibited Users

Except as specifically permitted in Sections 1,2 and 3 above, you agree not to (i) use, modify, merge or sub-license the Software or any other of Aladdin's products except as expressly authorized in this Agreement and in the HASP HL Protection and Licensing Guide; and (ii) sell, license (or sub-license), lease, assign, transfer, pledge, or share your rights under this License with/to anyone else; and (iii) modify, disassemble, decompile, reverse engineer, revise or enhance the Software or attempt to discover the Software's source code; and (iv) place the Software onto a server so that it is accessible via a public network; and (v) use any back-up or archival copies of the Software (or allow someone else to use such copies) for any purpose other than to replace an original copy if it is destroyed or becomes defective. If you are a member of the European Union, this Agreement does not affect your rights under any legislation implementing the EC Council Directive on the Legal Protection of Computer Programs. If you seek any information within the meaning of that Directive you should initially approach Aladdin.

5 Limited Warranty

Aladdin warrants, for your benefit alone, that (i) the Software, when and as delivered to you, and for a period of three (3) months after the date of delivery to you, will perform in substantial compliance with the HASP HL Protection and Licensing Guide, provided that it is used on the computer hardware and with the operating system for which it was designed; and (ii) that the HASP(r) key and the HASP(r) HL key for a period of twelve (12) months after the date of delivery to you, will be substantially free from significant defects in materials and workmanship.

6 Warranty Disclaimer

ALADDIN DOES NOT WARRANT THAT ANY OF ITS PRODUCT(S) WILL MEET YOUR REQUIRMENTS OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR-FREE. TO THE EXTENT ALLOWED BY LAW, ALADDIN EXPRESSLY DISCLAIMS ALL EXPRESS WARRANTIES NOT STATED HERE AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NO ALADDIN'S DEALER, DISTRIBUTOR, RESELLER, AGENT OR EMPLOYEE IS AUTHORIZED TO MAKE ANY MODIFICATIONS, EXTENSIONS, OR ADDITIONS TO THIS WARRANTY. If any modifications are made to the Software or to any other part of the Product by you during the warranty period; if the media and the Product is subjected to accident, abuse, or improper use; or if you violate any of the terms of this Agreement, then the warranty in Section 5 above, shall immediately be terminated. The warranty shall not apply if the Software is used on or in conjunction with hardware or program other than the unmodified version of hardware and program with which the Software was designed to be used as described in the HASP HL Protection and Licensing Guide.

7 Limitation of Remedies

In the event of a breach of the warranty set forth above, Aladdin's sole obligation shall be, at Aladdin's sole discretion: (i) to replace or repair the Product, or component thereof, that does not meet the foregoing limited warranty, free of charge; (ii) to refund the price paid by you for the Product, or component thereof. Any replacement or repaired component will be warranted for the remainder of the original warranty period or 30 days, whichever is longer. Warranty claims must be made in writing during the warranty period and within seven (7) days of the observation of the defect accompanied by evidence satisfactory to Aladdin. All Products should be returned to the distributor from which they were purchased (if not purchased directly from Aladdin) and shall be shipped by the returning party with freight and insurance paid. The Product or component thereof must be returned with a copy of your receipt.

8 Exclusion Of Consequential Damages

The parties acknowledge, that the Product is inherently complex and may not be completely free of errors. ALADDIN SHALL NOT BE LIABLE (WHETHER UNDER CONTRACT, TORT (INCLUDING NEGLIGENCE) OR OTHERWISE) TO YOU, OR ANY THIRD PARTY (INCLUDING, WITHOUT LIMITATION, YOUR DISTRIBUTORS AND USERS OF YOUR SOFTWARE PROGRAM) FOR ANY LOSS OR DAMAGE (INCLUDING INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES), INCLUDING, WITHOUT LIMITATION, ANY LOSS OR DAMAGE TO BUSINESS EARNINGS, LOST PROFITS OR GOODWILL AND LOST OR DAMAGED DATA OR DOCUMENTATION, SUFFERED BY ANY PERSON, ARISING FROM AND/OR RELATED WITH AND/OR CONNECTED TO ANY USE OF THE SOFTWARE AND/OR ANY COMPONENT OF THE PRODUCT, EVEN IF ALADDIN IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

9 Limitation Of Liability

IN NO EVENT SHALL ALADDIN'S TOTAL MONETARY OBLIGATION AND LIABILITY, UNDER ANY CLAIM FOR ANY CAUSES OF ACTION PURSUANT TO THIS AGREEMENT, EXCEED THE PAYMENTS MADE BY YOU TO ALADDIN FOR THE PRODUCT/S THAT GAVE RISE TO THE ACTION OR CLAIM, AND IF NO SUCH PRODUCT/S ARE SO APPLICABLE THEN ALADDIN'S LIABILITY SHALL NOT EXCEED THE AMOUNT OF FEES PAID BY YOU TO ALADDIN HEREUNDER DURING THE TWELVE (12) MONTHS PRECEDING THE EVENT.

10 No Other Warranties

Except as specifically provided herein, Aladdin makes no warranty or representation, either express or implied, with respect to its products as described in the preamble of this agreement, including its quality, performance, merchantability or fitness for a particular purpose.

11 Termination

Your failure to comply with the terms of this Agreement shall terminate your license and this Agreement. Upon termination of this Agreement by Aladdin: (i) the license granted to you in this Agreement shall expire and you, upon termination, shall discontinue all further use (including without limitation sub-licensing) of the Software and other licensed Product(s); and (ii) you shall promptly return to Aladdin all tangible property representing Aladdin's intellectual property rights and all copies thereof and/ or shall erase/delete any such information held by it in electronic form. Sections 1, 4, 6, 7, 8, 9, 10, 11 and 12 shall survive any termination of this Agreement.

12 Governing Law & Jurisdiction

This Agreement shall be construed and governed in accordance with the laws of Israel (except for conflict of law provisions) and only the competent courts of Tel-Aviv, Israel shall have jurisdiction in any conflict or dispute arising out of this Agreement. The application of the United Nations Convention of Contracts for the International Sale of Goods is expressly excluded. The failure of either party to enforce any rights granted hereunder or to take action against the other party in the event of any breach hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights or subsequent actions in the event of future breaches.

13 Third Party Software

If the Product contains any software provided by third parties, such third party's software is provided “As Is” without any warranty of any kind and shall be subject to any and all limitations and conditions required by such third parties.

14 Miscellaneous

This Agreement represents the complete agreement concerning the license hereunder and the subject matter hereof and may be amended only by a written agreement executed by both parties. If any provision of this Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable.

I HAVE READ AND UNDERSTOOD THIS AGREEMENT AND AGREE TO BE BOUND BY ALL OF THE TERMS.

Certifications

CE Compliance

The HASP product line complies with the CE EMC Directive and related standards*. HASP products are marked with the CE logo and a HASP CE conformity card is included in every shipment or upon demand.

*EMC directive 89/336/EEC and related standards EN 55022, EN 50082-1.

FCC Compliance

FCC authorities have determined that HASP is not a Class B Computing Device Peripheral and therefore does not require FCC regulation.

UL Certification

The HASP product line successfully completed UL 94 Tests for Flammability of Plastic Materials for Parts in Devices and Appliances. HASP products comply with UL 1950 Safety of Information Technology Equipment regulations.

ISO 9001:2000 Certification

The HASP product line is designed and manufactured by Aladdin Knowledge Systems, Inc., an ISO 9001:2000 certified company. Aladdin's quality assurance system is approved by the International Organization for Standardization (ISO), ensuring that Aladdin products and customer service standards consistently meet specifications in order to provide outstanding customer satisfaction.

Certificate of Compliance

Upon request, Aladdin Knowledge Systems, Inc. will supply a Certificate of Compliance to any software developer who wishes to demonstrate that the HASP product line conforms to the specifications stated. Software developers can distribute this certificate to the end user along with their programs.

Contents

Part I HASP HL Orientation

Part II HASP HL Protection

Part III HASP HL Licensing

Part IV Distributing HASP HL Software

List of Tables

About this Guide

The *HASP HL Software Protection and Licensing Guide* is designed to help software publishers protect and license their software using the HASP HL system. The guide is designed to provide background information and details on how the HASP HL system can best serve your protection and licensing requirements.

This guide is divided into four parts.

Part I — HASP HL Orientation — introduces the HASP HL system, presents basic protection and licensing concepts, and leads you through the process of setting the system. You should read this part after opening your HASP HL Developer's or Starter's kit.

Part II — HASP HL Protection — contains an overview and an in-depth presentation of the HASP HL protection methods. The section includes strategies for maximizing the protection of your software using HASP HL software. This section is aimed at software developers interested in using either of the HASP HL protection methods to protect software.

Part III — HASP HL Licensing — details how to effectively use HASP HL software to license your protected software. This section is particularly relevant to product and business managers within a software publishing enterprise. It is also relevant for the operations staff and anyone else involved in production. If you plan to license your software, we strongly recommend that you read this section to see how HASP HL can best be used to meet your licensing requirements.

Part IV — Distributing HASP HL — details HASP HL software that can be delivered to end users to ensure optimal performance of protected software together with the HASP HL hardware device. This part also describes the various ways and means of effectively delivering each available HASP HL software component.

Throughout this guide noteworthy comments, suggestions, cautions and warnings are displayed in special framed inserts containing the following symbols:

Symbol	Meaning
	Caution or warning
	Noteworthy comment and tip
	Useful idea or suggestion to enhance use or performance of the HASP HL system

For your convenience, a comprehensive glossary provides concise explanations of HASP HL terms.The glossary is available on page 207.

Information on technical specifications, the HASP HL API reference and a troubleshooting guide are also included in this guide.

Chapter 1
Introduction

Welcome to HASP HL!
This chapter includes the following topics:

- An introduction to the HASP HL protection and licensing system, its features and its advantages
- Introduction to the HASP HL range of hardware and software
- Description of the contents of the HASP HL Developer and Starter Kits

About HASP HL

HASP HL is a hardware-based protection and licensing system that is easy to use and very reliable. HASP HL prevents the unauthorized use of software, protects software copyright and intellectual property, and offers multiple licensing models. The system provides you — the software vendor — with complete control over the use of your software and supports innovative sales models, thus increasing software revenue.

When using a HASP HL-protected application at run time, application queries the HASP HL connected to the computer. If the response returned by the HASP HL is as it should be, and if the licence stored inside the HASP HL is valid, the application executes. If the response is incorrect, the application may not load, may switch to a demo version, or certain features are not made available to the user.

Implementing HASP HL security and licensing is easy, yet the level of security it provides is extremely high. Once your application is protected, it can be activated only when the HASP HL key originally supplied with your software is connected to the computer.

Advantages of the HASP HL System

Substantial Flexibility

The HASP HL system provides the widest range of products, solutions, and features in the software protection industry, including memory, network and time-based solutions for multiple hardware platforms.

Licensing using HASP HL is handled independently from protection. HASP HL provides tools that enable you to update and modify the licensing terms of your protected applications.

Maximum ease-of-use

HASP HL Envelope is a quick and easy way to implement HASP HL protection for your software using a friendly graphical user interface.

A short learning curve and a standard application programming interface (API) for all products, ensures easy and rapid incorporation of the HASP HL software into the application. HASP HL ToolBox is provided as an interactive method for using and studying the HASP HL API. The API is independent of the HASP HL model used or the platform running the protected application. This greatly simplifies the protection process.

Support for a wide range of programming environments

The HASP HL system includes interfaces and samples for numerous compilers and programming languages, enabling a quick and effortless implementation of HASP HL.

Support for a wide range of operating systems

Supported operating systems are: Windows 98 SE/ME/2000/ XP/Server 2003, Mac OS X and Linux.

Cross-platform solution

HASP HL provides a platform-independent solution. One HASP HL key can be used to protect Windows, Mac and Linux applications, thus saving development time in integrating protection, and reduces shipping, logistics and other costs.

State-of-the-art security

AES-Based Encoding Capabilities in the Hardware

The Advanced Encryption Standard (AES) algorithm is the basis for HASP HL encryption and decryption. The data encoding and decoding facilities incorporated in the HASP HL hardware facilitate a close integration of the hardware with the software being protected. Intelligent functions within the key itself allow critical functions of the protected software to be dependent on the presence of the correct key, otherwise they cannot operate correctly.

When using a publicly known algorithm like AES to encrypt data, the secret is kept within the encryption key and not in the algorithm. Furthermore, our hardware ensures that this secret 128 bit encryption key never leaves the HASP HL key. Every HASP HL customer is assigned a unique encryption key or keys.

Advanced Protection Algorithms and Anti-Debugging Technology

HASP HL software uses state-of-the-art code protection algorithms, plus the most advanced anti-debugging technology in the industry. Special anti-hacking features implemented in the HASP HL software create practically impenetrable obstacles for would-be hackers.

Scrambled Communication

To enhance security, all communication between the application and the HASP HL is randomly scrambled. This inhibits any emulation of the HASP HL hardware key.

Automatic Licensing

HASP HL enables you to define and implement multiple licensing schemes. Within a single tool — HASP HL Factory— you can define and apply licenses to HASP HL keys. You do not have to code your licensing terms within you application source. Checking which modules are licensed to run, for how long, how many times, how many users etc., is all executed automatically by the HASP HL system without requiring any developer intervention.

Benefits to Your Customers

HASP HL protection benefits both you and your customers. The following are a few reasons why software protection is beneficial to your customers.

Cost-Effective Software

As HASP HL protection increases your sales and revenues, you can budget more financial resources to the ongoing development and support of your product. In turn, your paying customers can enjoy more advanced products, faster product development and also higher-quality technical support.

Protecting the License Agreement

Software protection helps maintain the integrity of your software license agreement. HASP HL is the least painful, least intrusive means of assuring compliance with the license agreement. It ensures that customers do not have to police their own employees or risk violating the license agreement.

Protecting Investments of Legitimate Users

HASP HL protects legitimate users from the unfair practices of dishonest users, who do not pay for the software they abuse and waste your valuable technical support resources.

Worldwide Service and Support

Multiple Production Facilities

Production facilities on four continents ensure rapid and punctual supply, with back-up capabilities if necessary.

Local Service and Support in Over 40 Countries

With 9 international offices and over 40 local distributors, support for HASP HL is available virtually whenever and wherever required.

Aladdin Consulting

For more detailed advice and training on HASP HL implementation issues, contact our team of international consultants. They can provide you with tailored training sessions on the following:

- Integration of HASP HL into your product
- Analysis of the best protection strategy for your applications
- Assistance in implementing your protection and licensing designs

HASP HL Protection Keys

HASP HL keys are available for connection to a USB port and include various models listed in Table 1.1.

Table 1.1 HASP HL Models

HASP HL Model	Memory Size	Number of Licenses	Highlights
HASP HL Basic	none	1	encoding/decoding
HASP HL Pro	112 bytes	16	encoding/decoding HASP HL ID
HASP HL Max	4 KB	112	encoding/decoding HASP HL ID
HASP HL Time	4 KB	112*	encoding/decoding HASP HL ID real-time clock
HASP HL Net 10	4 KB	112	encoding/decoding HASP HL ID network access
HASP HL Net 50	4 KB	112	encoding/decoding HASP HL ID network access
HASP HL Net 250+	4 KB	112	encoding/decoding HASP HL ID network access
*8 reserved for expiration date			

All HASP HL keys are cross-platform USB devices that are used to protect applications running on Windows, Mac and Linux platforms.

HASP HL Models

HASP HL Basic– Low cost, high security

HASP HL Basic is the most cost-effective software protection device we offer. Using all the state-of-the-art techniques included in the HASP HL system, this hardware model offers an extremely secure, yet surprisingly low-cost solution for your protection needs.

HASP HL Pro and HASP HL Max – The most versatile and secure software protection keys available

These two models combine the inherently high level of encryption-based security with the flexibility of up to 4KB of secured read/write memory and a unique ID number for each key. HASP HL Pro and HASP HL Max enable in-the-field upgrading capabilities that let you implement your marketing strategy by enforcing sales models such as subscription, demo, try-before-you-buy and rental for up to 112 applications – all using a single key.

HASP HL Time– Software Protection with a Real-Time Clock

HASP HL Time contains an internal real-time clock, indicating the exact time (hours, minutes and seconds) and date (day, month, year). Specifically designed to enable software renting or leasing, it also lets you charge clients periodically for software use and maintenance. HASP HL Time contains 4KB of secured read/write memory and a unique ID number. HASP HL Time supports up to 104 licenses - it reserves 8 licenses for setting expiration dates.

HASP HL Net - Providing Licenses in a Network

This is the ultimate software protection solution for various network environments. Connecting a single HASP HL Net to any network station, protects your application, and controls the number of stations using it simultaneously. HASP HL Net has a unique ID, 4KB of memory and can store up to 112 licenses.

There are three types of HASP HL Net:

- HASP HL Net 10 — support for up to 10 concurrent users
- HASP HL Net 50 — support for up to 50 concurrent users
- HASP HL Net 250+ — support for at least 250 concurrent users.

HASP HL Developer Kit (DK)

The HASP HL Developer Kit contains software and hardware to enable you to evaluate HASP HL protection and licensing.

Software

The HASP HL software is contained on a single CD-ROM.

Hardware

The HASP HL Developer Kit includes the requested DEMO hardware keys based on your order.

The DEMO key should only be used for evaluation purposes. When you order HASP HL, you are assigned a unique Vendor Code. Use the DEMOMA together with the DEMO to evaluate HASP HL protection and licensing software.

HASP HL Starter Kit (SK)

The HASP HL Starter Kit is similar to the HASP HL Developer Kit. The main difference is in the type of HASP HL hardware included. This hardware is unique to your company and can only be used in conjunction with your Vendor Code. This code is contained in the Master HASP HL that is also included in the kit.

The HASP HL keys provided in the HASP HL Starter Kit can be used to protect and license software, and can be distributed to your end users. Protect your software and then simply order the number of keys you require.

Chapter 2
Key Concepts in HASP HL

This chapter introduces key concepts behind the HASP HL system. It includes the following topics:

- Your unique HASP HL codes & keys
- What makes the HASP HL system unique?
- How does the HASP HL system work?
- HASP HL protection and licensing
- Sample HASP HL workflow

To ensure that you use the HASP HL system effectively, we recommend that you familiarize yourself with the concepts and terms described in this chapter.

Your Unique HASP HL Codes & Keys

The HASP HL keys that you order from Aladdin contain unique information that is specific to your company. This information is used by the HASP HL system to communicate with your keys, and also to differentiate your keys from those belonging to other software vendors.

The vendor-specific keys and codes that you receive from Aladdin are listed on the next page.

- **HASP HL Demo:** A fully functional HASP HL key shipped in HASP HL Developer's kits. Used for evaluation purposes only. All HASP HL Demo keys are assigned the DEMOMA Batch Code.The corresponding DEMOMA Vendor Code and other identifiers are integrated within all HASP HL tools. When evaluating the HASP HL system with a HASP HL Demo key, you do not need a Master HASP HL key.
  DO NOT DISTRIBUTE SOFTWARE PROTECTED WITH A HASP HL DEMO KEY!
- **Master HASP HL:** A special HASP HL key that contains the unique codes and identifiers assigned to you by Aladdin that are used with the HASP HLsystem. This key must not leave your possession. It is required to extract the Vendor Code data used by Vendor Center tools for protecting and licensing software.
- **Batch Code:** Five to eight characters representing your unique Vendor Code. It is printed on the label of each HASP HL, and on the Master HASP HL that is associated with your batch of keys.Specify your Batch Code whenever you place future orders for HASP HL keys from Aladdin. When you order a new Batch Code from Aladdin, you will also receive a corresponding Master HASP HL.
- **Vendor Code:** A unique code assigned by Aladdin that is extracted from the Master HASP HL. The code contains information required by the HASP HL software to communicate with your HASP HL keys. The Vendor Center tools help you extract the Vendor Code from the Master key.

The keys described above are usually delivered at part of the HASP HL Developer or Starter kits. For more information, refer to "HASP HL Developer Kit (DK)" (page 8).

Figure 2.1 illustrates the relationship between the Master HASP HL keys and the HASP HL system.

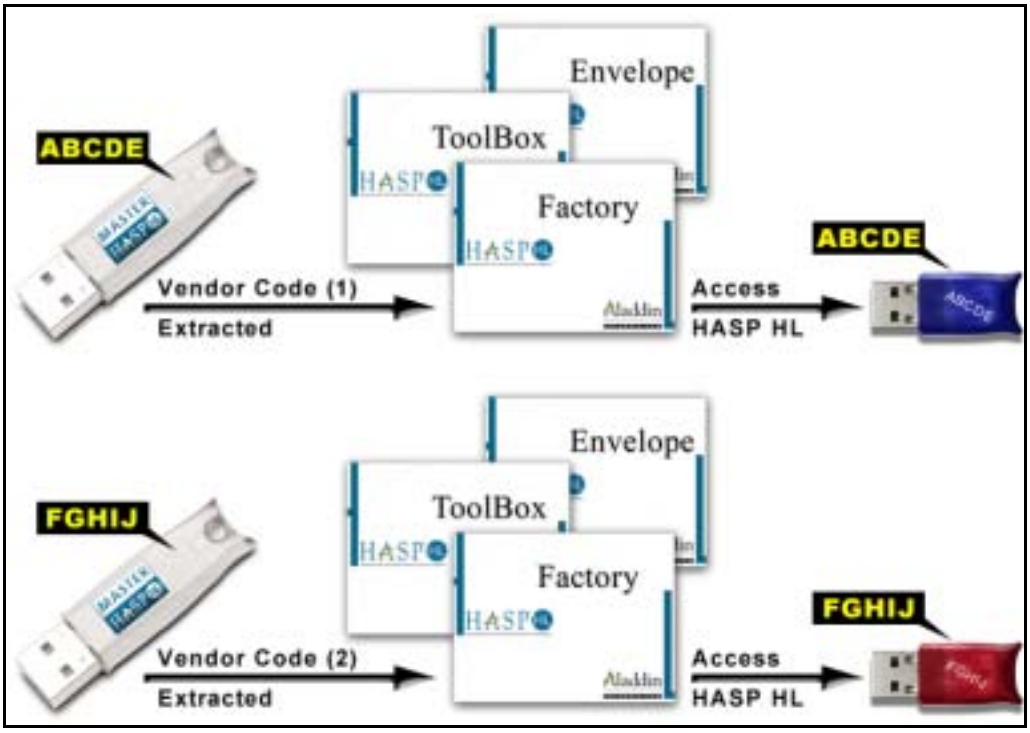
Figure 2.1 Master HASP HLs and keys of Vendors A & B

When you place your first order for HASP HL keys, you are assigned a unique Vendor Code which is coded into the Master HASP HL. In Figure 2.1, Vendor A is assigned Batch “ABCDE” and is provided a Master HASP HL with this Batch Code. All HASP HL keys shipped to this vendor with contain and be marked with this code.

Meanwhile Vendor B is assigned a different Batch Code, “FGHIJ”, and this is the way all HASP HL keys earmarked for this vendor are marked. Aladdin always ships your first order of HASP HL keys together with a corresponding Master HASP HL.

How is the Vendor Code derived from the Master HASP HL? In Figure 2.2, Vendor Code 1 — assigned to Software Vendor A — is extracted from Master HASP HL key of Batch Code “ABCDE”. Vendor A uses any of the Vendor Center tools to extract the Vendor Code. Vendor Code 1 is used by the HASP HL system to access all HASP HL keys of Batch Code “ABCDE”. Meanwhile Vendor Code 2 — assigned to Software Vendor B — is extracted for Master HASP HL of Batch Code “FGHIJ”. Vendor Code 2 is used by the HASP HL system to access all HASP HL keys of Batch Code “FGHIJ”.

Figure 2.2 Extracting Vendor Codes for Vendors A and B



What makes the HASP HL system unique?

Protect Once-Deliver Many™

This concept lies at the heart of the HASP HL protection and licensing system. At the heart of this concept, the process of protecting your software is completely separated from the process of defining sales and licensing models.

Protection

Protection is the process that involves securing the application and incorporating security strategies which are defined and performed by your development team. With HASP HL this process is performed once. Your development team need not be concerned about business aspects such as sales models and how to license the protected application. For a graphical illustration, refer to stage one in Figure 2.3 (page 23).

Licensing

Defining the sales models and licensing terms is an ongoing process that involves decisions on how the software is sold, licensed and distributed. This process is undertaken by product managers or sales and marketing managers, and does not influence or affect the process of protecting the software. With the HASP HL system, product managers are completely independent of your development team. Whenever there is a need for a new license model, they can define and implement it without involving your development team.

The separation of protection and licensing is reflected within the HASP HL system by providing separate tools to execute these two processes. Developers need to only once apply protection to the application, and disregard licensing issues. They indicate whether automatic licensing should be invoked by the HASP HL system, but do not actually implement it – this is done automatically by the HASP HL system. Subsequently the protection process is much quicker, more intuitive and focused.

Product managers don't have to depend on their product development team each time they need to introduce a new sales model. They simply use specifically designed HASP HL tools that enable them to define how the application is licensed. They define license templates and intuitively generate licenses and license upgrades that are used with the Remote Update System. This gives them more flexibility and freedom in defining new sales models and the ability to quickly respond to new business requirements.

For a graphical illustration on the separation of protection and licensing in the HASP HL system, refer to Figure 2.3 (page 23).

How does HASP HL protection work?

Applications linked to HASP HL Hardware

The basis of the HASP HL system is the strong inherent link that is formed between the protected application and its corresponding HASP HL.

Protection is based on making access to the protected application dependent on the presence of the correct HASP HL.

From a run time perspective, HASP HL protection involves a simple process:

a. The protected application queries the HASP HL connected to the computer.

b. If the response returned by the HASP HL is as it should be, the application executes.

c. If the response is incorrect, the application may not load, may switch to a demo version, or may limit certain features.

Identifying the HASP HL Key

HASP HL keys contain information specific to your company, thus ensuring the uniqueness of the keys you distribute together with your protected software.

The presence of the appropriate HASP HL can be verified by any of the following methods:

- Using the hardware-based encryption engine
- Checking for the key-specific ID
- Using the key's memory functionality

Using the Encryption Engine

When you implement HASP HL protection, you check for the presence of the required key. The HASP HL system performs these checks by encoding and decoding data using the AES-based encryption engine within the HASP HL.

Verifying the presence of the HASP HL using the data encoding and decoding functions, requires a certain amount of planning. You must encode data in advance. This data is sent to the key at run time for decoding.

The decoded data can be verified by using the data in your protected application. Refer to "Encode/Decode Data with the HASP HL Key" (page 90) for more information. After the data is decoded, it is used by the protected application. The encoded data is a function of the data sent to the HASP HL, together with the Vendor Code. Thus, encoding the same string with two different codes leads to different results.

To encode data, either use HASP HL ToolBox or the HASP HL API. For more information, refer to "HASP HL ToolBox" (page 57) or to "Implementing the HASP HL API" (page 59).

Using the HASP HL Memory Options

All HASP HL keys — apart from HASP HL Basic — contain internal read/write memory. You can use the HASP HL memory to do an of the following:

- Control access to different software modules or different software packages.
- Assign a unique code to each software user.

- Use the memory to store licenses from your own licensing schemes.
- Save passwords, program code, program variables, and any other data.

For information on the read/write memory available with different HASP HL models, refer to Table 1.1 (page 6).

You can edit the memory of a key using either the HASP HL ToolBox or the HASP HL Factory tools. For more information, refer the online Help system of either tool.

Checking the HASP ID Number

Each HASP HL key has a unique HASP ID number. A protected application can check and verify this ID number.

HASP ID numbers let you distinguish between the users of your application. By checking for the HASP ID number in your application you can decide how to respond if a specific HASP HL is or is not present.

All keys apart from HASP HL Basic have an ID number.

HASP HL Protection Methods

HASP HL offers two different protection methods:

- HASP HL Envelope
- HASP HL application programming interface (API)

HASP HL Envelope

The simplest and quickest way to protect an application is to use the HASP HL Envelope. HASP HL Envelope provides very effective and powerful protection. HASP HL Envelope adds a protective shield around executable files and DLLs. The HASP HL Envelope also scrambles your file, and incorporates HASP HL checks and advanced anti-debugging features. After protecting your application with HASP HL Envelope, it cannot run without the correct HASP HL.

As the HASP HL Envelope does not require access to the application source code, it is a quick and easy method of protection. At the same time, it provides a very high level of protection, making it virtually impossible to debug or disassemble your protected software.

For information on using HASP HL Envelope, refer to "HASP HL Envelope Protection" (page 69).

HASP HL Application Programming Interface (API)

If you have the source code of the application you want to protect, you can link the HASP HL to your application by using the HASP HL API.

Use the API to insert calls to the HASP HL throughout your application. With the API, you can check for the presence of the HASP HL key whenever you choose, and decide how to respond when the correct HASP HL is not connected. In addition, you can check a HASP HL memory key for sensitive data you may have stored in its memory.

Which Method to Use

You can use either the HASP HL Envelope or the API, or combine both protection method.

Use the HASP HL Envelope when you want quick and easy protection, or when you do not have access to the source code.

Use the API when you have access to the source code and when you want to customize your protection by implementing how and when calls are sent to the HASP HL by the protected application.

Both the HASP HL Envelope and the API are very powerful protection methods. Protecting your application with only one protection method ensures a high level of security. However, we recommend you implement both Envelope and API protection if possible. Each method has its unique features, and complements and enhances the other. For more information, refer to "HASP HL Protection - Best Approaches" (page 86).

HASP HL Licensing

As mentioned earlier in this chapter, licensing and protection are independent processes in the HASP HL system. You can licence multiple features or applications using a single HASP HL. To review the available licensing capabilities of the different HASP HL models, refer to Table 1.1 (page 6).

Licensing Options

With HASP HL you can control the usage of your protected software by:

a. Setting the number of activations for a particular feature or an entire application. This is particularly useful when supplying demo versions of your software.

b. Setting an expiration date for a particular feature or an entire application. This feature is particularly useful when leasing and renting your software.

c. Setting the number of stations which can run the program concurrently with a network. This is particularly useful for large enterprise-wide versions of your software.

How does HASP HL Licensing work?

A HASP HL-protected application performs several checks.

a. The protected application first determines whether or not the correct HASP HL is connected to the computer.

b. If the correct HASP HL is present, a further check is undertaken to see if the application or feature is authorized to run. This authorization is based on the terms of the license for the feature or application.

If the application is authorized to run then further specific checks are performed, depending on the model of HASP HL protecting the application:

- With HASP HL Max and HASP HL Pro, a check is executed to see that the number of authorized activations has not been exceeded. With each activation of the application, the number of authorized activations is decreased by one. When the application starts and the number of authorized activation equals 0, the protected application aborts and the relevant error message is displayed.
- With HASP HL Time, the expiration date listed within the key’s memory is checked and compared to the HASP HL

Time's real-time clock. If the expiration date has passed, the feature or application aborts with an error message.

- If you are using HASP HL Net, the protected application accesses the memory of the key to check if the number of stations allowed to simultaneously run the application has not been exceeded. For more information, refer to "HASP License Manager" (page 151).

Remote Update System

Licenses are applied to HASP HL keys using HASP HL Factory. However, through the HASP HL system's unique Remote Update System, you can also generate files for your end users to apply license updates without requiring them to remove the keys from the field. All HASP HL keys, except the HASP HL Basic, can be updated remotely. HASP HL Basic has no licensing features.

The remote update system also enables you to receive information on the current license status of deployed HASP HL keys. For more information, refer to "Remote Update System" (page 127).

HASP HL Workflow

This section presents a sample workflow for the Protect Once-Deliver Many concept to clarify the way the HASP HL system works.

A software vendor wishes to protect their word processing application. Following discussion within the business development unit, it is determined that three specific features should be licensed:

- PRINT
- SAVE
- IMPORT

The HASP HL system provides specific tools for the software vendor's business unit, its developers, and operations staff to execute the workflow that follows.

1. The business unit, having decided which application components to license, assigns a Feature ID to each licensed component. A list of features, shown below, is created and handed to the software development department

Table 2.1 Sample Feature List

Feature	Feature ID
PRINT	34
SAVE	4
IMPORT	48

2. Based on the feature list provided by the PM, the developer implements protection. The developer knows only the Feature IDs and needs not be concerned with any commercial or licensing issues, nor which type of HASP HL key (model) is to be shipped with the application.
   For example, the developer incorporates the HASP HL API to invoke automatic licensing for Feature ID number 34. The developer does not care whether the eventual software users are charged for the print feature according to the number of times it is invoked, whether it is limited by an expiry date, or whether it is limited to a specific number of concurrent users in a network environment. These are issues that are determined by the business unit and are irrelevant at this stage of implementing protection. The developer simply implements protection using Feature IDs as part of the protection parameters.
3. The business unit determine sales packages and licensing terms. A package contains one or more features and the licensing terms for each of the features. Table 2.2 lists sample packages for the workflow described in this section, the features they contain, and their specific licensing terms.

Table 2.2 Sample List of Packages

Package Name	Included Features	Licensing Terms	HASP HL Model Used
DEMO	PRINT SAVE IMPORT	30 DAYS for each feature	HASP HL Time
Standard	PRINT SAVE	Unlimited	HASP HL Pro
Enterprise	PRINT SAVE IMPORT	Unlimited	HASP HL Pro
Professional	PRINT SAVE IMPORT	Limited to 25 concurrent users	HASP HL Net

4. The business unit can now instruct sales to begin to sell the product and to receive orders. The orders/production department receives orders for specific packages. With a click of a button, the production person selects the appropriate package and connects the desired HASP HL, which is programmed accordingly.

5. At a later stage, the customer can request updates. These updates can be extensions to existing licenses or licenses to use additional modules. The orders/production person receives information about the required update from the customer, and through a simple process, selects the appropriate package. This package is transformed into update data that is sent to the customer.

HASP HL accommodates customers who do not wish to use the licensing workflow outlined in this section. In this case, developers simply have to implement protection, whereby the protected applications access a "default feature" provided by the system. The default feature is always available so long as the required HASP HL key is connected.

Figure 2.3 Protect Once-Deliver Many Sample Workflow

Chapter 3
Setting up HASP HL

This chapter is designed to help you get started using the HASP HL system. The following topics are included:

- A brief overview of available HASP HL software
- HASP HL software protection - a quick tour
- How to install HASP HL on your system
- How to install the HASP HL device drivers
- Introducing Master HASP HL keys into your system

Available HASP HL Software

Use the HASP HL installation CD to install the following on your system:

- Drivers and daemons used to communicate with HASP HL keys.
- The **Vendor Center** suit of programs used to protect and license your software.
- The HASP HL API and its libraries are used to integrate protection within your software. Samples are included for various programming languages.
- HASP License Manager used to communicate between the protected application and HASP HL Net network keys.

- Aladdin Monitor used to track usage of protected applications in a network environment.
- Aladdin DiagnostiX and Aladdin DiagnostiX Memory Beamer used to troubleshoot problems with deployed keys.

HASP HL Drivers

The HASP HL device drivers serve as the main link between the HASP HL and the protected application once it is deployed at the customer's site. End users need to have the appropriate device drivers installed in order to run the protected application.

HASP HL device drivers can run on the following operating systems: Windows 98SE/ME/2000/XP/Server 2003, Mac and Linux.

You can also integrate the HASP HL driver installation into the installation of your protected application. For Windows XP and Windows Server 2003, the HASP HL drivers are available via Windows Update.

Vendor Center

The Vendor Center contains the following programs:

- **HASP HL Envelope** — applies security to your software executable files within a protective shield.
- **HASP HL Factory** — used to define and produce licenses for your protected application, and to initialize the memory in HASP HL keys.
- **HASP HL ToolBox** — used to learn the HASP HL API and to generate code to include within your software's source code.

HASP HL API

The HASP HL API application programming interface (API) enables you to protect your application by inserting calls to the HASP HL key throughout your source code.

The root directory on the HASP HL installation CD is subdivided according to operating systems, and each system folder contains a Sample folder. The **Sample** folder contains samples for various compilers and programming languages: Each sample includes:

- Libraries that must be linked to your application
- A sample application that demonstrates the use of the API.

HASP License Manager/Aladdin Monitor

Use both utilities when the protected application is linked to a HASP HL Net operating in a network environment.

End User Utilities

These include Aladdin DiagnostiX and the Aladdin DiagnostiX Memory Beamer. For more information, refer to "Diagnosing HASP HL Keys" (page 189).

HASP HL Software Protection - A Quick Tour

The following section is a quick tour of the steps required to protect an application with HASP HL. As you will see, your software can be protected within a matter of minutes. To run this tour you will need the following:

- Your HASP HL installation CD
- The HASP HL Demo provided in the Developer kit sent to you by Aladdin.

1. Install HASP HL Software

Insert the HASP HL installation CD and follow the on-screen instructions until you receive a prompt stating that the installation was successful.

2. Connect Hardware

Connect the HASP HL Demo key to the USB port on your computer. This key should only be used for evaluation purposes.

3. Launch Vendor Center

From Windows **Start**, select the following path from the program menus:

`Programs/HASP HL/Vendor Center`

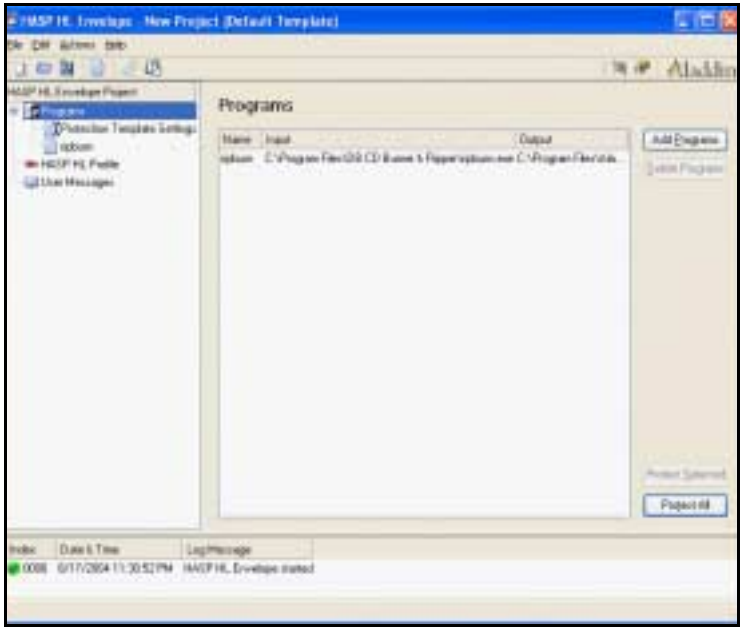
The Vendor Center window show in Figure 3.1 is displayed.

Figure 3.1 HASP HL Vendor Center

4. Select HASP HL Envelope

Figure 3.2 is displayed. HASP HL Envelope is a tool used for quick automatic protection.

Figure 3.2 HASP HL Envelope Window



5. Select input for protection

a. Click the **Programs** icon in the **HASP HL Envelope Project** Pane.

b. Select an executable file on which to apply HASP HL protection. Simply drag and drop the file into the **Programs** pane shown above.
OR

You can click Add Programs in the **Programs** pane of HASP HL Envelope.
The **Add Programs** dialog shown in Figure 3.3 is displayed. Use the dialog to point to the file you want to protect.

Figure 3.3 Add Programs dialog

Select the file you want to protect and click **Open**.The dialog is closed. The program you have selected is listed in the **Programs** pane.

c. To protect the file, click **Protect All** in the **Programs** pane.
A status bar is displayed:
.

When the file protection process is over the following message is displayed.

Congratulations!!! You have protected the program using HASP HL technology. Click **OK** to close the message.

7. Test your protected output

a. Launch the protected application by running it with the HASP HL Demo attached to your computer.

b. Close the protected application.

c. Unplug the HASP HL Demo.

Re-launch the protected application. The following error message is displayed.

The protected application cannot run if the required HASP HL is not present!

Installing HASP HL under Windows

The first step in setting up your HASP HL system is to install all the necessary software. This section explains how to install the HASP HL software under the Windows operating system. The HASP HL developer environment and Vendor Center tools can be installed on Windows 2000 and newer Windows platforms.

Installing the HASP HL Software

Insert the HASP HL Installation CD in your CD-ROM drive. The HASP HL Setup Wizard automatically starts and displays the **Welcome** screen shown in Figure 3.4.

Figure 3.4 HASP HL Installation Wizard

Follow the on-screen instructions to install the contents of the CD.

If for any reason the installation does not automatically start, run *setup.exe* from the Windows directory on the CD.

Installation Setup Structure

A typical installation on Windows would contain the following folders in the HASP HL directory:

- **API** — containing the API libraries.
- **Drivers** — containing the HASP HL device drivers. For more details, refer to the next section below.
- **Samples** — containing sample applications for various programming languages and interfaces.
- **Vendor Tools** — contains the Vendor Center tool suite and the Aladdin Monitor.
- **Docs** — contains an online version of this guide plus supplementary documentation for using the HASP HL system.
- Redistribute — contains the software to be deployed at the end user's site. For detailed information, refer to "HASP HL Software for End Users" (page 137).

The setup structure described above mirrors the setup of the HASP HL installation CD under the `Windows/Installed` directory.

Installing the HASP HL Device Drivers

The HASP HL device drivers are installed automatically when using the general HASP HL software setup described in the previous section.

You can also install the HASP HL device drivers separately using the *haspdinst.exe* or the *HASPUserSetup.exe* applications. These applications are located in Drivers directory of your HASP HL installation CD.

The *haspdinst.exe* and *HASPUserSetup.exe* utilities recognize the operating system in use and install the correct driver files to the required location.

Administrator privileges are required to install the HASP device drivers under Windows 2000/XP/ Server 2003.

The HASP HL drivers load dynamically as soon as the HASP HL is connected.

The *HASPUserSetup.exe* application

This is a GUI-based installation program to separately install the HASP HL drivers on Windows 98/ME/2000/XP/Server 2003 systems. The file is located in the following directory on the HASP HL installation CD.

```
Windows/Installed/Drivers
```

To launch the application, click *HASPUserSetup.exe* and follow the on-screen instructions.

The *haspdinst.exe* Utility

The *haspdinst.exe* utility is a command-line application that installs the HASP HL device drivers under Windows 98SE/ME/2000/XP/Server 2003 systems.

To install the HASP HL Device Drivers:

Type `haspdinst -i` in the command line.

A message is displayed informing you that the HASP HL device drivers were successfully installed.

To remove the HASP HL Device Drivers:

Type `haspdinst -r` in the command line.

A message is displayed informing you that the HASP HL device drivers were successfully removed.

To upgrade a HASP HL Device Drivers:

Install newer HASP HL device drivers according to the installation procedure described above. The *haspdinst.exe* utility automatically handles the upgrading process.

Other Haspdinst Utility Commands

Table 3.1 lists other commands that can be executed with the *haspdinst.exe* utility.

Table 3.1 Haspdinst Commands

Command	Description
-info	Displays the installation status
-h and -?	Displays a list of the available commands
-kp	Enables the installation program to 'kill' all processes accessing the drivers
-cm	Sets the installation program to only display critical messages (e.g. instructions to reboot)
-fr	Sets the installation program to remove driver by force
-nomsg	Sets program to display no messages

Installing HASP HL under Mac

The HASP HL software for Mac platforms is located in the *MacOS* directory on the HASP HL installation CD. The folder includes the following sub-folders:

- **API** — contains header files and libraries for various programming languages.
- **Docs** — contains electronic versions of HASP HL documentation.
- **Redistribute** — contains HASP HL software that you can distribute together with your protected software and HASP HL keys.

This folder also contains the files you need for installing the HASP HL daemon.

- **Samples** — contains sample programs for various programming languages implementing the HASP HL API.
- **VendorTools** — contains a command-line version of the HASP HL Envelope. For more information on using this application, refer to "HASP HL Envelope for Mac Applications" (page 79).

The contents of these folders are not automatically installed on your computer. Please copy the software you need from the HASP HL installation CD.

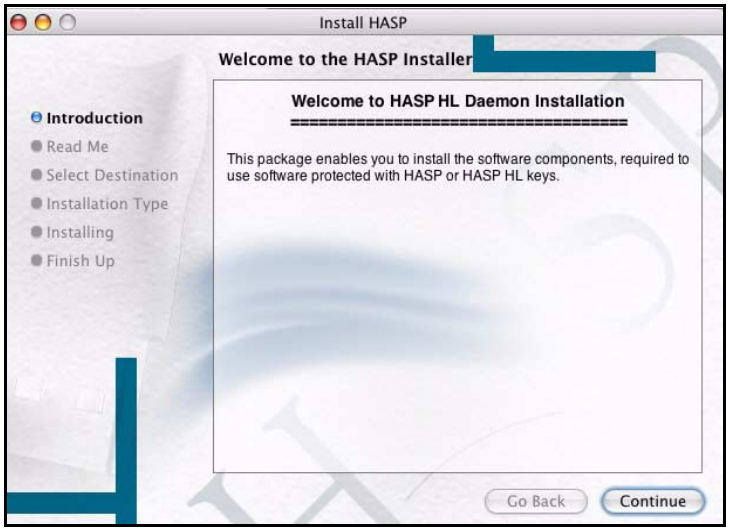
Installing the HASP HL Daemon

To enable your system to recognize and communicate with HASP HL hardware you need to install *aksusbd* — the HASP HL daemon.

Installing the daemon

1. Mount the *HASP Installation.dmg* install image, located at:
   `MacOS/Redistribute/Runtime/dmg`
   A volume containing 4 files is mounted.
2. Open the *AKSUSB Install.pkg* file.
   The built-in installer, shown in Figure 3.5, leads you through the installation process.

Figure 3.5 HASP HL Mac Installer Start screen



The above installation requires the BDS- Subsystem to be installed on your machine. If this is not installed, use the *AKSUSB Install_forced.pkg* file described in step 1. This option requires a reboot once the installation is completed.

Installation Checklist

The following items should be installed on your machine before attempting to use the daemon:

a. The daemon should be located the following volume: `/usr/libexec/`

b. Two startup files ensuring that the daemon is activated after reboot should be located in the following volume: `System/Libraries/StartupItems/Aladdin/`

The LED on the HASP HL hardware should be lit when all the items are correctly installed.

Options for *aksusbd*

General Modifications

You can modify the behavior of the daemon by using the command-line switches listed in Table 3.2.

Modification process

To temporarily modify the behavior of the daemon:

1. Launch **Activity Monitor** located in `/Applications/Utilities/`
2. Select the **aksusbd** process from the list displayed when viewing **All Processes**.
3. Terminate the process — **Admin** privileges are required to execute this command.
4. Open a shell window and start the daemon using the modified options.

To ensure that your modified settings are applied after reboot, you should add your modifications to the Aladdin shell script located in:

`/System/Libraries/StartupItems/Aladdin/`

Table 3.2 Command Line Parameters for *aksusbd* (Mac)

Switch	Meaning
-v	Print version number as decimal, format xx.xx.
-l <value>	Select type of diagnostic messages. Possible values are: 0 - only errors 1- normal (default) 2 - verbose 3 - ultra verbose The messages are logged in syslog with priority kern.info (and kern.debug). Refer *to /etc/syslog.conf* to see where the messages are stored. Usually it is kept in: */var/log/messages*.
-u <umask>	Specifies the permission bits for the special socket file. Default is 666 (access for all users).
-d	Sets a time delay in milliseconds for accessing the HASP HL once it is connected. The default value is 250.
-h	Displays the help for the command line application.

Installing HASP HL under Linux

The HASP HL software for Linux includes the following:

- A HASP HL daemon
- HASP HL Demo application and the source code
- HASP HL libraries

The Linux software is located in the *Linux* directory on the HASP HL installation CD.

The simplest way to install HASP HL for Linux is to use RPM packages that are provided on the HASP HL installation CD. HASP HL software updates for various Linux distributions are constantly updated and readily made available at:

`http://www.hasp.com/downloads.`

For detailed information on the individual components, refer to the *readme.txt* files in the *Linux* directory.

Installing the HASP HL Daemon

To access the HASP HL key, the *aksusbd* daemon has to be loaded.

All actions described in this section should be executed as root.

Installing *aksusbd* using RPM packages

RPM packages are available for various distributions of SuSE and Red Hat. These packages enable a quick and automated installation process.

Installing HASP HL Daemon using RPM packages:

1. Select the package you require from the *Linux* directory.
2. Enter the installation command using the following format:
   `rpm -i [package name]`

For a complete listing of all available commands available for the RPM packages, use the following command:
`man rpm`

Aladdin Daemon Installation (*aksusbd*)

Enabling Access to USB Keys

To enable access to USB keys, *subdues* must be mounted on */proc/bus/us*. On newer distributions, it is mounted automatically (e.g SuSE 8.0).

To mount *usbdevfs* manually, use the following command:

```
mount -t usbdevfs none /proc/bus/usb
```

Loading the Daemon

Load the daemon by launching it with the following command:

```
<path>/aksusbd
```

The daemon will fork and put itself in the background.

A status message is generated in the system log informing you whether the installation was successful or not. It reports the daemon version, and the version of the HASP HL API used for the USB.

If */proc/bus/usb* is not mounted when launching *aksusbd*, USB keys cannot be accessed.

Is is recommended that the daemon be started at system boot with some script located in:

/etc/rc.d/init.d or */etc/init.d*

Options for *aksusbd*

Table 3.3 Command Line Switches for *aksusbd* (Linux)

Switch	Meaning
-v	Print version number as decimal, format xx.xx.
-l <value>	Select type of diagnostic messages. Possible values are: 0 - only errors 1- normal (default) 2 - verbose 3 - ultra verbose The messages are logged in syslog with priority kern.info (and kern.debug). Refer *to /etc/syslog.conf* to see where the messages are stored. Usually it is kept in: */var/log/messages*.
-u <umask>	Specifies the permission bits for the special socket file. Default is 666 (access for all users).
-d	Sets a time delay in milliseconds for accessing the HASP HL once it is connected. The default value is 250.
-h	Displays the help for the command line application.

Extracting Vendor Codes

Once you order HASP HL keys from Aladdin, you will be assigned a unique Batch Code. Your HASP HL keys will arrive together with a Master HASP HL. For a review of the relationship between the batch and your keys, refer to "Your Unique HASP HL Codes & Keys" (page 9).

The Master HASP HL you were provided with, contains your unique Vendor Code and other important information that is required when using the HASP HL system to protect and license software.

To extract the information you need from the Master HASP HL key:

1. Connect the Master HASP HL to your computer.
2. Start any of the **Vendor Center** tools (**HASP HL Envelope**, **HASP HL Factory** or **HASP HL ToolBox**).
3. The tool detects the new Master HASP HL.

There is an option to specify an **Access Code** for the Master HASP HL. To specify a code, click **Change**. In the displayed screen, enter a code in the fields provided. You will need to provide this code when using HASP HL Factory to license your protected software. Click **OK** to close.

4. Select a filename under which to save the vendor code information. It is strongly recommended that you store all the Vendor Codes in:
   Program Files/Aladdin/HASP HL/Vendorcodes/
   The Vendor Center tools search this directory by default.

Chapter 4
Protecting Software

This chapter is an overview of HASP HL protection. It includes the following topics:

- A summary of how HASP HL protection works
- Fundamental aspects of HASP HL protection
- Introduction to HASP HL protection methods

The following chapters discuss the different HASP HL protection methods in detail, and how to maximize software protection using the HASP HL system.

"HASP HL API Protection" (page 53) describes the HASP HL API method in detail.

"HASP HL Envelope Protection" (page 69) provides a detailed description of HASP HL Envelope.

Tips and strategies to maximize software protection using the HASP HL system, are covered in "Protection Strategies" (page 85).

HASP HL Protection

HASP HL is an innovative and advanced hardware-based solution for protecting software against illegal or unauthorized use. The system hinders illegal access or execution of protected programs.

A deployed HASP HL-protected program requires access to a specific HASP HL. The protected program queries the key for particular information. If the HASP HL is not present, or the information returned is suspect, the program either does not load or stops functioning.

Once you have selected a HASP HL protection method, implementation is straightforward. Regardless of the protection strategy selected, protected programs only work properly if they can access the information stored in a specific HASP HL.

Elements of HASP HL protection

The HASP HL protection system is based on the following:

- Identifying the correct HASP HL
- Protecting programs and data files
- Utilizing the AES encryption engine
- Confidential protection parameters
- Anti-debugging and reverse engineering
- Using the HASP HL memory

Identifying the HASP HL Key

The HASP HL, or to be more precise — the intelligence contained within the key — is the hardware component of the HASP HL protection system.

Regardless of the protection method adopted, protected programs only function when they can access the required information contained in a specific key. A HASP HL together with its ‘intelligence’ cannot be cloned to replicate the link between the hardware device and the protected program.

The main factor governing HASP HL protection is whether a deployed program can identify and access the intelligence contained in a specific key at run time. This factor is unambiguous and clear: the key is either available or not!

HASP HL Protection Methods

HASP HL offers two main protection methods:

- HASP HL API
- HASP HL Envelope

When protecting programs using either of these methods, you are essentially forming an inherent link between the protected program and a specified HASP HL.

What can be protected?

The HASP HL system offers a wide choice in what to protect. You can apply protection directly to:

- Compiled executables or DLLs.
- Specific features or entire programs — the HASP HL system protects all levels of software. HASP HL protection can be applied at the function level or throughout an entire program.
- Sensitive data and intellectual property.

All the above are protected against any attempt at reverse engineering.

The protection parameter options available in the HASP HL system are detailed in the next two chapters.

AES Encryption

The protected program relies on the ‘intelligence’ of a specific HASP HL in order to function. In addition to the checks for the key, data can be encrypted and decrypted using the intelligence available in the HASP HL.

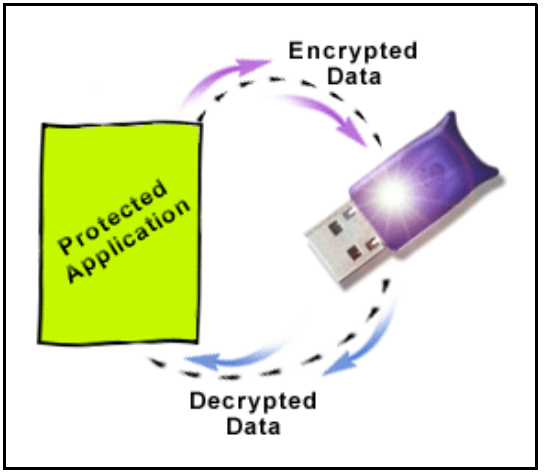
AES Encoding and Decoding

The encryption engine in the HASP HL hardware is based on the AES algorithm. HASP HL encryption uses a set of confidential 128-bit encryption keys that remain in the HASP HL.

Your protection schemes should always involve greater sophistication than merely confirming the presence of the required HASP HL. However, verifying the required HASP HL through data encryption and decryption needs a certain amount of planning. First, encoded data must be available. This data must then be sent to the key, where it is decoded.

If the data is correct, the HASP HL is considered to be “present.” This protection is graphically illustrated in Figure 4.1. For more information, refer to "Encoding Functions" (page 66).

Figure 4.1 Encoding/Decoding Data



Confidential Protection Parameters

The essence of software protection is confidentiality. Without confidential elements, any software security system is rather exposed.

Vendor Code

Each HASP HL customer has a unique Vendor Code. This code must be kept confidential.

The Vendor Code forms an integral part of the protection parameters that constitute the inherent link between the protected programs and the HASP HL. However, the Vendor Code is only part of the link, and not the entire link. The code on its own, is not enough to enable illegal use of the software. It merely provides the protected software with access to the key and its resources.

All HASP HL protection tools require the Vendor Code. For information on how to access the code, refer to "Extracting Vendor Codes" (page 44).

Exploiting the HASP HL Memory

Most HASP HL models have memory chips. HASP HL memory can be utilized (read and write) to form part of the protection scheme for the software. Confidential data can be stored, including parts of program code, customer name, or any other data in the HASP HL memory.

Use the memory editors included in HASP HL ToolBox to read or write data in the HASP HL memory. For more information, refer to "Memory Functions" (page 66).

Anti-Debugging and Reverse Engineering Measures

The HASP HL system protects intellectual property and provides the capability of combatting anti-debugging and reverse engineering. These measures usually try to unravel the protection scheme of protected software by tracing the compiled application to its source code. HASP HL Envelope implements contingency measures to ward off such attacks and prevent hackers from uncovering algorithms used inside protected software.

Selecting a Protection Method

As mentioned earlier in the section "HASP HL Protection Methods" (page 47), HASP HL offers two software protection methods: the HASP HL API and the HASP HL Envelope. Both methods establish an inherent link between the protected software and the intelligence contained in a specific HASP HL.

When choosing a protection method, the following should be taken into consideration:

- What should HASP HL protect?
- How should the HASP HL protection parameters be applied?
- Time factors — is an automated speedy solution required?
- Is flexibility in implementing the protection scheme important?

What to protect?

When protecting software with HASP HL, there are various options for applying protection. HASP HL API is used to protect the software before it is compiled. Protection can also be applied after the software is compiled using the HASP HL Envelope. You can chose whether to apply protection to an entire program, a subprogram, or simply to a feature.

How to apply protection

When using the HASP HL API, protection is integrated at the source code level in a carefully considered manner. You determine where in the source code to place calls to the HASP HL API. The HASP HL Envelope offers a more automated way of protecting software. You define settings for protection parameters that are applied to protected programs.

Is control over the protection scheme important?

When applying protection using the HASP HL API, you control the entire protection process. You determine when the protected program queries the HASP HL key, and how it should react in different scenarios. With the HASP HL Envelope, protection is more random when wrapping the compiled programs. When the HASP HL Envelope is run twice to protect the same program, two different output files are produced with different protective modules and shields.

HASP HL API versus HASP HL Envelope Implementation

You may opt to use either protection method, or both, depending on your specific requirements. Use Table 4.1 to determine which method best meets your particular requirements.

Table 4.1 HASP HL Envelope versus HASP HL API

HASP HL Envelope	HASP HL API
• Automatic protection process. Define specific protection parameters for your programs. • Quick and easy protection process that shields your software. • No source code required. • Anti-debugging and reverse engineering measures provided.	• Manual implementation of calls to the HASP HL API. • Controlled and deliberate process ensuring maximum security. The strength of protection is proportional to how much of the API's functionality is invested in implementation. • Source code must be available. • Maximum flexibility.

Chapter 5
HASP HL API Protection

This chapter describes the HASP HL API protection method. It includes the following topics:

- An overview of the HASP HL API
- Prerequisites for using the API
- Learning to use the API
- Implementing the API
- Available API functionality

Overview

The HASP HL application programming interface (API) is a robust method of software protection, the strength of which is wholly dependent on how you chose to implement it. To fully unleash the protection offered by the HASP HL API, strive to maximize the complexity and sophistication of your implementation.

The HASP HL API Reference (page 215) details the functions that make up the HASP HL API. Refer to this appendix for specific information on particular functions.

To protect your software via the HASP HL API, insert calls to a HASP HL throughout your application's source code. You can check for the presence of the HASP HL at any point of the application run time, and designate responses to these queries. If the key is not found, you could have the protected application suspend or terminate itself.

Using the HASP HL API, you can also check the memory of a HASP HL key for specific data, and encode or decode particular data.

The extent to which the functionality afforded by the HASP HL API is utilized, determines the overall level of software security. It is therefore essential, that before protecting application, you are familiar with the overall functionality of the API.

To facilitate a speedy learning curve, we also recommend that you test and familiarize yourself with specific API functions by using the HASP HL ToolBox tool. Also check the HASP HL API Sample folders for your specific compiler. Each HASP HL interface includes a sample application demonstrating API usage, together with a specific header file. The sample applications are located in the **Sample** folder within the Windows, Mac OSX and Linux directories on the HASP HL installation CD.

Universal API

It is worth highlighting a specific point about the HASP HL API: the HASP HL API is a universal API that works in tandem with all HASP HL models and at run time on all major platforms.

Regardless of the HASP HL model you use, this factor is transparent as far as the API implementation or usage is concerned. There are specific API functions that are only relevant for particular HASP HL models, however the implementation of protection is independent of the type of keys linked to your protected applications.

Utilization of the API is independent of the type of access mode used to search for a particular HASP HL. The same API functions are used to set programs to access remote keys, or keys that are connected locally.

HASP HL API implementation is not dependent on the HASP HL model or the platform in which the protected application is deployed

The HASP HL API is also universal in terms of the run time environment in which the protected applications are deployed. HASP HL is a cross platform system supporting Windows, Mac and Linux operating systems.

Prerequisites for API Usage

Before using the HASP HL API, the HASP HL device driver or daemon must first be installed. Refer to Installing the HASP HL Software (page 33) for information on how to install the required drivers or daemons.

Platform requirements

The HASP HL API is designed to work on:

- Windows 98SE and newer Windows platforms
- Mac OSX
- Linux

Vendor Code

It is necessary to provide the Vendor Code in order to access the HASP HL key and its resources including memory. Vendor Codes are usually stored in the *Vendorcodes* directory. On all Windows installations, the directory is located at: `Program Files/Aladdin/HASP HL/Vendorcodes.`

HASP HL customers are initially provided HASP HL Demo keys that work with the DEMOMA Vendor Code. This Vendor Code can be used to apply protection with the HASP HL API.

Do not distribute software protected with HASP HL Demo. This key is purely for evaluation purposes.

Once you are supplied a Master HASP HL key, you should extract the unique Vendor Code that has been assigned to you.

If you have already introduced your Master HASP HL key while using one of the **Vendor Center** tools, skip the section below.

Extracting the Vendor Code from the Master HASP HL

Follow the procedure below to extract all the vendor code information you need from the Master HASP HL:

1. Connect the Master HASP HL to your computer.
2. Start any of the **Vendor Center** tools (**HASP HL Envelope**, **HASP HL Factory** or **HASP HL ToolBox**).
3. The Wizard detects and lists all new Master HASP HLs.

You have the option to specify an **Access Code** for the Master HASP HL. To specify a code:

a. Click **Change**.

b. In the displayed screen, enter a code in the fields provided. You will need to provide this code when using HASP HL Factory to license your protected software.

c. Click **OK** to close.

4. Enter a filename under which to save the vendor code information. It is recommended that you store all the vendor codes in the *Vendorcodes* directory. On all Windows installations, the directory is located in:

`Program Files/Aladdin/HASP HL/Vendorcodes`
The Vendor Center tools by default search this directory

Learning the HASP HL API

The are two ways to study how the HASP HL API works and its range of capabilities.

- HASP HL ToolBox -is a tool with a graphic user interface that is part of the Vendor Center suite of programs.
- API Samples - to see an example of how to implement the HASP HL API, select the relevant sample folder according to your complier.

HASP HL ToolBox

To use the HASP HL ToolBox application you must have a HASP HL and a valid Vendor Code to access the key. The program is activated from the Vendor Center tool suite. For specific information on how to use HASP HL ToolBox, refer to the program's online Help system.

Access to the API

Essentially HASP HL ToolBox is an interactive interface to the HASP HL API. You execute calls to the API which are then relayed to the connected HASP HL.

HASP HL API Related Functionality

HASP HL ToolBox is a tutorial for the HASP HL API and offers the following:

- HASP HL ToolBox displays the source code generated for each function call. This generated source code can be copied and pasted in your application source code.
- Evaluation of the manual implementation of the HASP HL API. Every HASP HL API function included in HASP HL ToolBox is displayed on a separate screen. To execute a function call, you are required to provide specific information related to the chosen function.
- Memory buffers can be transferred to the AES encryption engine in the HASP HL key. The program can also be used to decrypt data buffers.
- The HASP HL ToolBox settings offers several programming language interfaces for the HASP HL API.

API Samples

Sample applications are provided to demonstrate how to implement HASP HL API protection in your source code. The samples provide simple examples of how the API functions work.

Your HASP HL installation contains folders for various interfaces and compilers. Each folder contains the requisite API libraries, a header file and the sample application. The HASP HL Demo key — marked DEMOMA —must be connected to your computer when using the sample applications.

Refer to the Aladdin web site and the HASP HL Installation CD for information on available samples for specific programming languages.

Implementing the HASP HL API

Planning your requirements

Before implementing the HASP HL API, some preliminary considerations must be made.

a. What do you want to protect? This may seem obvious but it is crucial in deciding where to place the calls to the HASP HL. Typically, you would want to verify the presence of the HASP HL key at startup. However, you can also identify certain aspects of the software to protect and apply your HASP HL API calls accordingly.

Feature protection is handled through the Feature ID parameter. You need to determine what program number to assign to the feature earmarked for protection. For more information see "hasp_login()" (page 233).

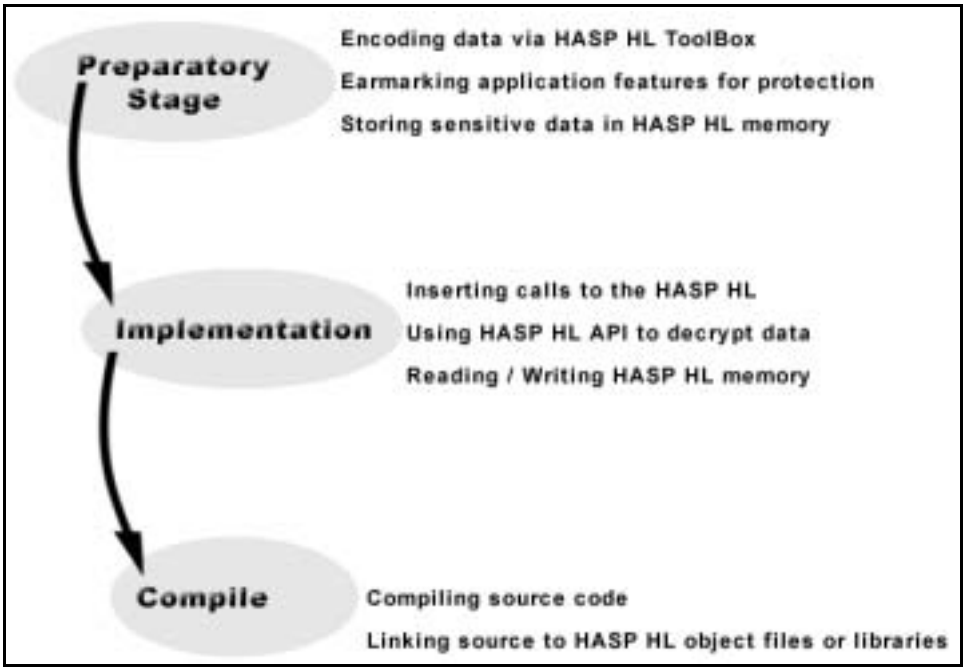
b. Will encoded data be included in my implementation scheme? If you plan to use encoded data at run time, use HASP HL ToolBox to encrypt the data. Once encrypted, insert the data when implementing the HASP HL API. This data is decrypted at run time by the HASP HL.

c. Is data going to be stored in the HASP HL memory? If the software is protected by a HASP HL key with memory functionality, sensitive data can be stored on the key. The HASP HL API enables access to read or write to the memory. Use HASP HL ToolBox to write data buffers to the HASP HL memory.

HASP HL API Implementation Stages

After taking the preliminary actions and decisions as outlined in the previous section, you are ready to protect your application with the HASP HL API. Figure 5.1 outlines the main stages in implementing the HASP HL API.

Figure 5.1 Stages of HASP HL API Implementation



To implement the HASP HL API:

1. Study the code of the sample application corresponding your development environment.
2. Insert a login call to the HASP HL key within your application source code.
   A successful login establishes a login session. The login session has its own unique handle identifier.

> The session identifier is self-generated and applies to a single login session. For more details, refer to hasp_login() (page 233).

3. Once a login session is established, you can use other HASP HL API functions to communicate with the key. For example, you can use the decrypt function to decode important data used by your application. You can also read data stored in the HASP HL memory, or set time stamps etc.
4. Using the output generated in Step 3, check for potential mismatches and issue the appropriate notification to the user.
5. Repeat steps 2-4 throughout the code.
6. Compile the source code.

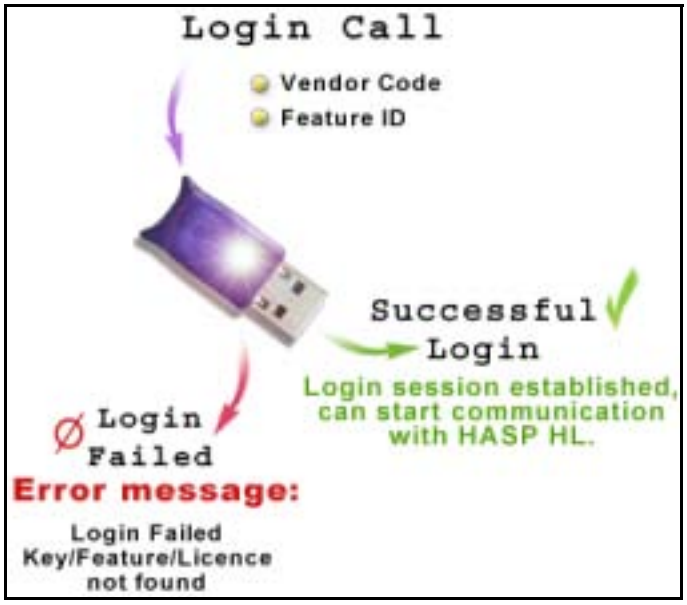
Once you have complied the source code, use the HASP HL Envelope to add an extra layer of protection to the executable. This also prevents reverse engineering of the protected code.

The HASP HL API Login Function

The login function is the gateway to HASP HL API implementation. To search for and communicate with a HASP HL, a successful login session must be established. Figure 5.2 reviews the operation of the login function within the context of implementing the HASP HL API. To log into a HASP HL, you need to provide a Feature ID and a valid Vendor Code.

Only after a successful login call is executed, other functions can be used to communicate with the key. If the HASP HL is not detected or connected to the computer, an error message is displayed. An error message is also displayed if the declared Vendor Code is not valid for the connected HASP HL.

Figure 5.2 HASP HL Login Operation Summary



Login Options

Like any other aspect of HASP HL API implementation, login calls are not dependent on the platforms targeted at run time or on particular HASP HL models. However, when establishing login calls you must determine what it is that you are actually logging into. When logging in you must declare:

- If you are logging into a default or a specific feature.
- How to search for the HASP HL key.
- How the login counter should be handled.
- Whether to enable or disable connection to the HASP HL via a terminal server.

Declaring Feature IDs

You can either log into a specific or default feature stored in the HASP HL key. The default feature is assigned program number 0.

The Feature ID is the corresponding ‘PROGRAM NUMBER FEATURE’ that exists inside a HASP HL key. Specific features are assigned program numbers, whereas the default feature is always available so long as the required HASP HL key is connected. The specification of a program number is particularly important for licensing protected software. For more details, refer to hasp_login() (page 233).

When logging into a licensed feature, the protected application not only checks for the presence of the HASP HL, it also checks the terms of the license contained in the key. If the license is valid, the feature is enabled. For more information, refer to "Assigning identifiers to features" (page 113).

There are additional aspects of the login call that can be controlled when implementing the HASP HL API.

Search Options

The default search setting enables the protected application to search both the local computer and the network for the required HASP HL. You can however limit the HASP HL search option to:

- Only search the local PC for a connected HASP HL. To set this restriction include
  `HASP HL_PROGNUM_OPT_NO_REMOTE`
  in the login declaration.
- Only search the network for a connected HASP HL. To set this restriction include
  `HASP HL_PROGNUM_OPT_NO_LOCAL`
  in the login declaration.

Login Counter

When a HASP HL license is accessed in a HASP HL Net key, license usage is determined per workstation. However you can override this condition and instead have license usage counted per process. This implies that the license counter is decremented per process. To set this status include
`HASP_HL_PROGNUM_OPT_NO_PROCESS`
in the login declaration.

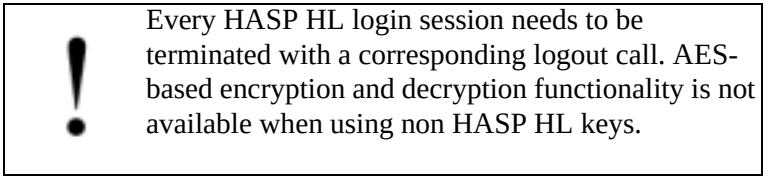
Terminal Server Detection

By default, the HASP HL system is designed to detect and prevent applications from running remotely on a terminal server. To override terminal server detection include
`HASP_HL_PROGNUM_OPT_TS`
in the login declaration.

Enable access to HASP3/HASP4 keys

By default, the HASP HL system does not enable access to earlier versions of Aladdin HASP keys. To override this restriction and enable access to HASP3 and HASP 4 keys, include
`HASP_HL_PROGNUM_OPT_CLASSIC`
in the login declaration.

Once you have logged into a HASP HL key and established a session, there is a wide range of HASP HL API functions that you can utilize in building a solid protection scheme. The next section discusses these options.



Available HASP HL API Functionality

The extent of protection afforded by the HASP HL API is dependent on the way that it is implemented. Calls to the HASP HL that are inserted within the code ultimately control access to the application at run time.

This section covers the HASP HL API options that are available once a successful login session is established. For a detailed discussion on how to optimize your HASP HL API implementation, refer to "Optimal HASP HL API Implementation" (page 89). For an demonstration of how the HASP HL API works, you should use HASP HL ToolBox. All functions listed below are detailed in the "HASP HL API Reference" (page 215).

Function groups

The available HASP HL API functions can be categorized into five groups based on common functionality and linkage.

- Session functions
- Memory functions
- Encoding functions
- Time functions
- Management functions

Session Functions

In order to implement most HASP HL API functions, you need to establish a successful login call to a license residing in a specific HASP HL key. The specific options available are described in "Login Options" (page 62). Every login call must be coupled with a corresponding logout call to end the login session.

In between logging in and logging out, other calls can be made to a specific HASP HL key as part of a login session.

Memory Functions

The HASP HL API can be used to:

- Read data buffers stored in the HASP HL memory.
- Write data buffers to the HASP HL memory.

The byte size of the data buffers is restricted by the memory available in the HASP HL model in question. Refer to Table D.2 "HASP HL Model Technical Specifications" on page 246 for the memory capacity available for specific HASP HL models.

Use the HASP HL memory to store information that can be used later to verify and identify the end user.

Use the memory to store data to be used by the application at run time. Access to this sensitive data forms an integral part of your protection scheme.

Encoding Functions

Once you have successfully logged into a HASP HL, you can encrypt or decrypt a data buffer using the AES based encryption engine in the key. The encryption engine uses symmetric encryption and therefore the same encryption key is used later to decrypt the data buffer.

Time Functions

If you are using a HASP HL Time key, HASP HL API can be used to access the real time clock in the key. This functionality enables you to:

- Set time
- Read time

Two date and time conversion functions are also included in the HASP HL API.

Management Functions

The HASP HL API includes functions that enable you to retrieve information on the current login session, the status of a deployed key, and license update.

Updates

You can use the HASP HL API to create updates using a special function call. Unlike the other HASP HL API functions this update function does not require a login session. For more information refer to "hasp_update()" (page 238). This function is the main facilitator of the Remote Update System.

HASP ID

You can specify the serial number of the HASP HL key — the HASP ID — to target actions such as updates or data retrieval to specific HASP HLs.

Chapter 6
HASP HL Envelope Protection

This chapter describes software protection using HASP HL Envelope. It includes the following topics:

- Prerequisites for using HASP HL Envelope
- Running HASP HL Envelope
- HASP HL Envelope protection features
- Handling data files for HASP HL Envelope

Functionality

The HASP HL Envelope is a wrapping tool that protects your applications within a secure shield. The tool offers advanced protection features to enhance the overall level of security of your software.

The HASP HL Envelope tool protects executables and DLLs. HASP HL Envelope protection provides a means to counteract reverse engineering and other anti-debugging measures.

By using HASP HL Envelope to protected your software, you establish a link between the protected software and the HASP HL. This link is broken whenever the protected software cannot access the required key, and the program ceases to function unless the HASP HL is made available.

To protect .NET assemblies use *dotnetenv.exe*. This is a command-line console program specifically designed to protect .NET assemblies. The program is located in the following directory on your HASP HL CD: *Windows\VendorTools\dot_NET_Envelope*

Implementing HASP HL Envelope protection is the fastest way to secure your software without requiring access to your software source code. The HASP HL Envelope is operated using a graphical user interface. The HASP HL Envelope graphical user interface enables you to:

- Protect programs.
- Define protection parameters.
- Specify a Vendor Code to authenticate the presence of a specific HASP HL key.
- Customize run time messages displayed for end users running protected programs.

In addition to linking protected programs to a specific HASP HL, HASP HL Envelope wraps the application file with numerous protection layers that are assembled randomly

The random multi-layer wrapping of protected applications by the HASP HL Envelope ensures that implemented protection strategies differ from one protected application to another.

Prerequisites for Using HASP HL Envelope

To use HASP HL Envelope, the software listed below must be available.

Software Requirements

All of the following components must be installed on your system:

- HASP HL drivers or daemons.
- HASP HL software including the HASP HL Vendor tools.
- A valid Vendor Code stored in the *Vendorcodes* folder. Refer to "Extracting the Vendor Code from the Master HASP HL" (page 56) for more details.
- *dfcrypt.exe* if you are planning to encrypt data files.
- Any of the following Windows operating systems: Windows 2000/XP.
- The executables you want to protect.

Running HASP HL Envelope

Use the following navigation path to launch the Vendor Center tool suite:
Start/Programs/HASP HL/Vendor Center.

Once the Vendor Center is activated, launch **HASP HL Envelope**.

Basic Protection Procedure

The procedure below describes how to protect applications using HASP HL Envelope. More detailed information on specific usage is provided in the HASP HL Envelope online Help system.

1. Launch **HASP HL Envelope** from the Vendor Center tool suite.
2. Designate what program(s) to protect.
3. Set protection parameters for the protected program(s).
4. Protect the program(s).
5. Distribute the protected software together with your coded HASP HL keys.

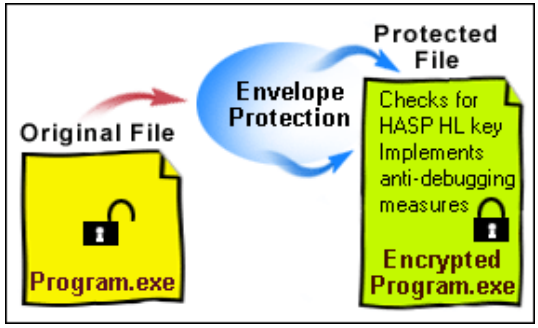
The HASP HL Envelope does not affect the files being protected. You should however designate a separate folder for the protected application in order to distinguish between source and output files.

HASP HL Envelope protection involves the application of protection parameters controlled by the engine running HASP HL Envelope. These protection parameters are applied to an unprotected source.

HASP HL Envelope does not affect the way the protected applications actually works. The only modification is that user access is conditioned on the presence of the required HASP HL. If the key is present, the protected file runs.

The logic of HASP HL Envelope protection is graphically illustrated in Figure 6.1.

Figure 6.1 HASP HL Envelope Operation



Running HASP HL Envelope in Command-Line

HASP HL Envelope can be initiated using a command-line prompt. This is useful when running automated processes that do not require a graphical interface. To access the command-line version of the tool, click *envelope.com*. This file is located in the following directory:

`Program Files/Aladdin/HASP HL/Vendortools/Vendorcenter`

The command-line version of the HASP HL Envelope is primarily used for automated processes. Create and save protection projects together with configuration files using *envelope.exe*, before running the command-line tool.

Starting the command line version

Type ENVELOPE in the command line. The system searches for and launches the *envelope.com* application.

Command- line options

The following parameters are available for use with the HASP HL Envelope command-line version:

Table 6.1 HASP HL Envelope Command-Line Parameters

Command	Description
-h /--help	Accesses the list of command-line parameters. After the help is displayed, you have to press **Enter** to return to the command-line console.
-p /--protect <configuration>	The command-line tool uses the specified configuration file as input data for the application-wrapping process. Configuration files contain definitions for protection parameters. Only a single file can be protected when using a configuration file.
-p /--protect <project>	The command-line tool uses the specified project as input data for the application-wrapping process — all the files included in the project are protected.
<project>	The command line version starts the GUI version with the specified project running as the current project.

HASP HL Envelope Protection Parameters

This section provides a general overview of the available protection settings.

HASP HL Envelope is an automatic method of applying HASP HL protection to your software. However, you can define and calibrate a range of protection parameters for the protected application. The settings are divided into two categories: mandatory settings, and preset settings affecting the attributes and the behavior of the protected file.

Mandatory Parameters

The following information must be provided in order to protect software using the HASP HL Envelope:

- Input File location — to protect an application its location must be specified.
- Vendor Code — you must provide a valid Vendor Code in order to access the connected HASP HL. On initial activation of HASP HL Envelope, the default Vendor Code is set as DEMOMA.

This is sufficient information to process the protection of an application. All other protection parameters use default values and are described below.

Optional & Preset Parameters

The optional parameters appear on three separate input screens in HASP HL Envelope. Each setting displays a default value that can be modified. These settings affect the attributes and behavior of protected applications. All the settings are detailed in the HASP HL Envelope online Help.

Searching for the HASP HL

HASP HL Envelope enables you to determine how the protected application searches for the required HASP HL.
The following options are available:

- Local and Network — the protected application first searches the local PC and then the network for the required key (Default).
- Locally — the protected application only searches the local computer for the required key.
- Network — the protected application only searches the network for the required key.

Protected Program Behavior

HASP HL Envelope enables you to define other properties for the protected application:

- How often random queries are sent to the HASP HL. These queries involve random encryption and decryption procedures.
- Time interval to check for the presence of the required key.
- Enabling and disabling support for programs that require overlays to execute properly.
- For licensing schemes involving concurrent usage, HASP HL Envelope enables you to decide whether to count activations per process or per station. For more information, refer to "HASP License Manager" (page 151).
- How long the protected application waits for the HASP HL device drivers to load.

Protection Attributes

You can set particular security attributes for the protected application. HASP HL Envelope includes parameters for:

- Detection of both system and user-level debugging measures. You can activate measures to be undertaken by the HASP HL system to block potential attacks that seek to undermine the protection scheme.
- Defining the number of protective module layers that are wrapped around the protected application. Possible values range from 1 to 50. The default setting is 12.

> Increasing the number of protective modules increases the startup time for a protected application and the resultant file size.There is also a trade-off between Encryption level and protected file size and startup speed. A higher encryption level causes a slower startup, and a larger protected application size.

- You can specify the frequency of HASP HL key access for scrambling. The setting controls the compactness of the HASP HL calls made by the protected application. An **Encryption Level** slider is provided to determine the frequency of HASP HL key access for scrambling.

Figure 6.2 Encryption Level slider

Data File Handing

Your protected software may require run time access to data files. HASP HL Envelope enables you to control the access of data files by the protected software. HASP HL Envelope offers the following control mechanisms:

- Data filters — you can determine what file types can be accessed at run time by the protected software. You can also determine what files to exclude.
- Data encryption keys — eight printable characters used to form an encryption key used to encrypt and decrypt data files.

Use the same encryption key when multiple applications access the same document set.

HASP HL Envelope includes the data filters and encryption key information as part of the protection scheme applied to the protected application. The encryption of data files is handled by the *dfcrypt.exe* command-line application. For more information refer to "Encrypting Data Files" (page 81).

Run time user support

You can customize run time messages for end users using applications protected by HASP HL Envelope. HASP HL Envelope includes a set of message codes. Each code is mapped to a corresponding message that is displayed at run time of the protected application. The displayed messages can be customized.

In addition you can choose to display a special message box for end users starting up a protected application. The message explains that there may be delays due to required data decryption.

HASP HL Envelope for Mac Applications

HASP HL Envelope protection can be implemented for Mac applications. HASP HL Envelope for Mac is an OS X terminal utility that enables you to protect Mach-O applications.

Protecting Mac Applications

You protect Mac applications by:

a. Defining and storing protection parameters in a configuration file.

b. Creating a copy of the file bundle containing the Mach-O executable.

c. Accessing the configuration file during the protection session.

The HASP HL Envelope configuration parameters are detailed in *Envelope Configuration Settings.pdf* which is available in the *Doc* folder of the HASP HL installation CD.

Once protection parameters are defined and stored in the configuration file, you activate the Envelope as follows:

hasphlmacho_darwin -c configuration.cfg

Override Settings for Configuration File

Several settings defined in the configuration file can be overwritten by the optional arguments listed in Table 6.2.

Table 6.2 Settings for HASP HL Envelope configuration file

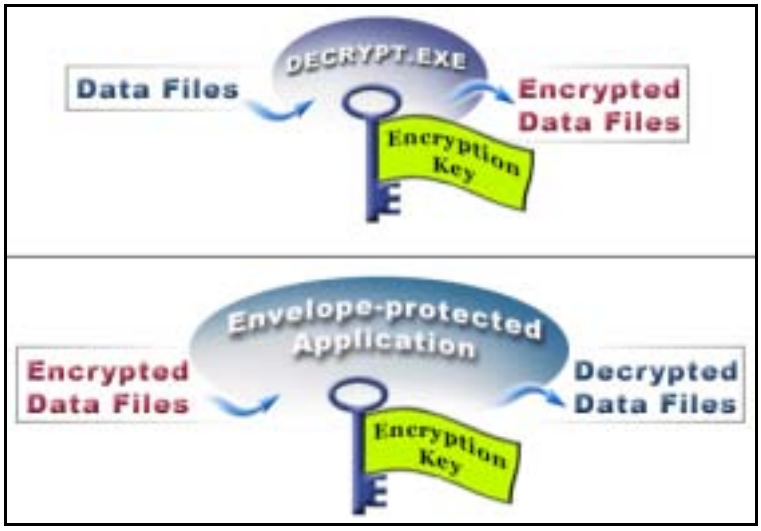
Setting	Description
-c <config filename>	Mandatory setting for the name of the configuration file.
-i <input filename>	Optional setting pointing to the Mach-O input executable contained in the application file bundle.
-o <output filename>	Optional setting pointing to the Mach-O output executable contained in the application file bundle.
-v <vendor code filename>	Optional setting for the Vendor Code used by the protected file to access the required HASP HL key.
-m <messages filename>	Optional setting for the file containing the definitions for messages displayed to end users at run time.

Encrypting Data Files

As mentioned in "Data File Handing" (page 77), applications protected with HASP HL Envelope can access encrypted data files defined by data filters and using a specified encryption key. HASP HL Envelope does not encrypt data files.

The *dfcrypt.exe* command-line application is used to encrypt data files. Dfcrypt generates data files that can be processed by executables protected by the HASP HL Envelope.

Figure 6.3 Data File Handling with *dfcrypt.exe*



Using *dfcrypt.exe*

This tool can only be used as a command-line application. It requires a particular set of input parameters to function. Once these parameters have been applied to a defined set of data files, the encrypted files can be accessed by HASP HL Envelope. Figure 6.3 (page 81) displays the operative procedures for data file handling using *dfcrypt.exe*. Once data files are encrypted, applications protected with HASP HL Envelope can decrypt and access these files. This is only possible, if at protection time, you define the encryption key in HASP HL Envelope.

dfcrypt.exe usage instructions

To use dfcrypt you need to specify the following:

- A command option
- A list of source files and directories
- A destination for the encrypted output

When specifying multiple source files or a directory, the destination input specified must be an existing directory.

The following format must be used to input parameters:

dfcrypt <option> source destination

For example:

```
dfcrypt -c:demoma.hvc -k:4873Asdb data.txt
data_crypt.txt
```

Available *dfcrypt.exe* commands

Table 6.3 below lists the available dfcrypt commands. An asterisk indicates mandatory commands.

Table 6.3 List of *dfcrypt.exe* command options

Command	Action
-e, --encrypt	Encrypts data, available by default.
-d, --decrypt	Decrypts data.
-c, --vcf:<file>	Specifies a HASP HL Vendor Code file.*
-k, --key:<key>	Specifies the encryption key to be used to encode data files. Should contain 8 printable characters.*
-o, --overwrite	Overwrites destination files.
-r, --recursive	Enables recursive handling of subdirectories.
-h, --help	Displays help screen listing dfcrypt commands.
-q, --quiet	Suppresses output by excluding copyright information and the progress indicator. Only error messages are displayed. This is particularly useful in Makefile integration.

Chapter 7
Protection Strategies

This chapter outlines strategies for maximizing HASP HL protection. It includes the following topics:

- Best approaches to overall HASP HL protection
- Optimizing HASP HL API protection
- Countering attacks though optimal software protection techniques
- Tips for maximizing the security of your protected application

The HASP HL system provides the best hardware and software protection tools available in the market today. However, it is up to you to take advantage of the benefits offered by HASP HL protection. The extent of protection is dependent on the way you choose to implement the protection scheme most suited for your requirements.

To achieve maximum protection with HASP HL, use both the Envelope and API protection methods.

If you would like special assistance in further strengthening your software security and protection, we can offer personalized support service from our team of Aladdin consultants. They can provide help on a wide range of issues, including protection strategies and implementation techniques.

For more information on consultation services offered by Aladdin, please contact your local Aladdin representative.

HASP HL Protection - Best Approaches

The two methods for implementing HASP HL protection have already been introduced and discussed in earlier chapters. Each protection method has its own merits, and different users have different requirements.

To decide which is the best protection method for you, review the different considerations presented in "HASP HL API versus HASP HL Envelope Implementation" (page 51).

When implementing software protection, it is recommended that you consider the following guidelines:

a. Use both the HASP HL Envelope and the HASP HL API to protect your software.

b. Do not rely on a simple immutable protection strategy.

Use Both HASP HL Methods

The best way to maximize protection is by using both HASP HL Envelope and HASP HL API to protect your software. Use the HASP HL API to implement calls to the HASP HL key, and after compiling your code, protect the executable with the HASP HL Envelope.

Once your source code is compiled, wrap your executables or dynamic libraries with the HASP HL Envelope.

Subsequent sections discuss the optimization of each protection method. However, bear in mind that using one protection method does not preclude the use of the other. This is especially true for users of the HASP HL API. It takes less than 5 minutes to apply HASP HL Envelope protection to your compiled executables!

Change Your Strategy

To achieve a high level of software security, you should modify your protection scheme often. Vary your methods, implementing different security measures for each version of your application. Upgrade your tools of defense regularly.

Aladdin is committed to supplying the best protection technology available on the market, and continually enhances its product line. Check the Aladdin web site periodically for information about new features in the HASP HL protection system.

A special link to the Aladdin site is provided in the Vendor Center tool suite. Use the link in the Vendor Center utility to get the latest software updates.

Contact your HASP HL representative for the latest updates on HASP HL software developments. Stay in step with Aladdin and always keep one step ahead of anyone who tries to attack your software.

Optimizing HASP HL API Protection

This section describes the type of software protection issues you should be concerned about, and offers protection tips and strategies. In general, because the HASP HL hardware is virtually impossible to break or duplicate, attacks are likely to focus on the software side — tracing the protection code and eliminating the protection schemes.

Software Protection Concerns

Most software publishers are concerned about two prevalent methods of attacking protected applications:

- Simulating calls to the protection device
- Simulating the software used by the manufacturer of the protection device

To emulate calls to the protection routine, an illegal user needs to tamper with the protected executable so that it does not send requests to the key, verify the results returned by the key, or act according to the results specified in the code. This kind of attack is used when it is assumed that protection has been poorly implemented.

To emulate the software of the key manufacturer, an attacker would usually manipulate the calls used to communicate with the hardware key. This ensures that calls to the key produce the 'correct' returns, even when the required HASP HL is not connected.

It is important to note that both of these methods are application-specific and cannot be applied as is to other protected applications.

To achieve maximum security, we recommend that you link your applications to HASP HL objects rather than DLLs. This enhances protection because objects are compiled as an integrated part of the executable output, making them more difficult to tamper with.

Optimal HASP HL API Implementation

This section provides tips and tricks on how to combat the attacks described in the previous section. Use as many of the tips as possible to maximize the security level for your software.

Keep in mind that when you implement protection, you should do so in a manner that considers possible implications for legitimate end users of the application. For example, a legitimate user may forget to connect the key to the computer. Your protection scheme should take this into account. As a rule, clever protection strategies should combat attacks on your software, not innocent users.

The following tips reflect optimal usage of the HASP HL API:

- Multiple calls to the HASP HL key
- Use the intelligence of the key to encode and decode data
- Implement the HASP HL API calls in a sophisticated manner
- Use the HASP HL memory

Use Many HASP HL API Calls

Inserting many calls to the HASP HL key exhausts those attempting to attack your protected software. A complex implementation of the HASP HL API, involving multiple calls to the HASP HL key, increases the difficulty in tracing and attacking your protection scheme.

The more calls and return codes that are checked by the HASP HL system, the more difficult it is to trace and remove all of them. The calls should also be made from as many different places within your code as possible.

Encode/Decode Data with the HASP HL Key

Do not only use a high number of HASP HL API calls in your code, but escalate the sophistication of the calls as well. A login call to a HASP HL that merely checks for the presence of the key is one of the simpler protection possibilities. Use the intelligence of the key to manipulate data in a way that hinders unauthorized attacks on your software.

Encoding data with the HASP HL AES-based encryption engine considerably enhances software security. By encrypting data used by your application, the decryption process is dependent on both the presence and the intelligence within the HASP HL.

By implementing a HASP HL API scheme in which data is decoded by a specific key, the association between the protected application and the HASP HL cannot easily be removed. Not only does the application have to be cracked, but also the data needs to be decoded.

The encryption and decryption processes are performed inside the HASP HL. This is well beyond the reach of any debugging tool.

There is no need to encode all data files handled by the protected application; instead you should focus on key data such as file headers, important constants in calculations, or specific fields in a database. You should consider encoding data that affects the main functions of the application.

Encoding process

For simple data sets, the basic encoding process is outlined below. These basic procedures can be modified to suit your specific requirements.

1. Connect your HASP HL to your computer.

Make sure you have the correct Vendor Code to communicate with the HASP HL!

2. Encrypt the data.

   Use HASP HL ToolBox to encrypt your data. For more information see the HASP HL ToolBox online Help.

3. Paste the generated code into your application source code.

   Replace the original clear text data in your application source with the include file or the encoded data.

   You can now decode the data on demand at run time within your application.

4. Use the HASP HL decrypt function to decode the encoded data using the connected HASP HL. For more information refer to "hasp_decrypt()" (page 220).

   You can now perform operations on this data. Remember to provide for error warnings for users when the required HASP HL is not present.

"HASP HL not found" and similar strings should not be encoded. These strings are displayed when the HASP HL is not connected; their proper decoding should not be dependent upon a response from the HASP HL key.

Avoid Repetitive Schemes

When you implement HASP HL API calls in the same manner throughout your code, it is easier to trace and understand the protection scheme. If an illegal user knows what to look for, protection is at risk.

Use the entire range of functionality offered by the HASP HL API but avoid repetitive steps.

Once your source code is compiled, the best way to introduce randomness into a protection scheme is to wrap your executables or DLLs using the HASP HL Envelope — implementing a different protection scheme for each executable file.

Split up HASP HL Verification

Verification of the HASP HL involves a three-phase process:

a. Using a valid Vendor Code and other parameters, a login call is made to a license within a specific HASP HL key.

b. The return values are evaluated.

c. A response is triggered according to the return values.

You should not code the above process sequentially, but rather split it up within your code. Separated steps are much more difficult to trace than sequential steps. Also, the triggered response to a missing HASP HL key should be delayed so as to surprise the user.

For example, you can check for the HASP HL when the user clicks on a specific menu item. Allow the user to work even though the key is not connected. Issue the message that the required key was not located only after an entirely different operation is executed. This way the user cannot easily establish a connection between a specific functionality and the check for the HASP HL.

Use a HASP HL with memory

When using a HASP HL with memory, you are vastly increasing the scope of protection possibilities offered by the HASP HL API. You can store data and values that are essential for your software to operate correctly.

You should use the HASP HL API memory functions to enable your program to read data stored in the HASP HL memory without checking for its validity. Only the ‘correct’ values enable the proper operation of the protected programs. It is hard to imagine someone correctly guessing the value required in order to enable the programs to function.

You can use the memory in a HASP HL to prevent a single key from loading your software on multiple terminals. Use the HASP HL API to generate and store a value in both the HASP HL memory and the RAM when the protected software launches. Periodically compare and verify that the two values are equal.

Smart Code Handling and the HASP HL API

This section outlines tips for handling software code together with your implementing of the HASP HL API. These are general tips and address situations that apply to all potential HASP HL users.

The tips included are:

- “Red herring” API calls
- Checksum code
- Using data stored in the HASP HL

Generate Noise through “red herring” API calls

Divert any potential attacker’s attention by encrypting data that has no bearing on the protected application. These “red herring” API calls cause distractions and pose as additional obstacles to anyone trying to trace and attack your protection scheme.

You can generate noise through random number generators, time values, intermediate results of calculations and other mechanisms. Of course, these calls to the HASP HL should not lead to any meaningful results and actions.

Checksum code

Performing a checksum enables you to determine whether or not the protected application has been tampered with.

To perform a simple checksum:

1. Calculate the checksum.
2. Compare it with the correct value. Make sure you separate the steps as outlined in "Split up HASP HL Verification" on page 92.
3. Continue running the application if the two values match, otherwise issue an error message.

Unfortunately, this technique is vulnerable to several attacks.

- The code can be patched and the check bypassed.
- It is possible force the checksum to return the correct value.

You can protect against this type of attack by avoiding simple modular addition or performing an XOR operation. Use CRC (cyclic redundancy check) or another algorithm sensitive to byte ordering.

Another technique is to avoid comparing the result of the checksum with a previously calculated value. Instead, use the checksum result to perform an action that results in an error if the wrong value was calculated.

For example, store the result as a variable and use that variable later on as a key to decode a certain code or data. This approach has the advantage of delayed reaction. In addition, the expected checksum value is not explicitly stored within the application.

Manipulate functionality as response to a missing key

There is a range of possible responses to trigger within the application's source code, when the required HASP HL is missing. The typical response is to insert a "HASP HL not found" message. However, this reveals that a check for the HASP HL was performed.

You should consider ways of disturbing the natural flow of the program as a response to a missing key. Unauthorized users may think that the disturbance is caused by a bug in the protected program. They should not realize that a check for a HASP HL check was performed, and that the problem is the deliberate consequence of the missing HASP HL.

For example, instruct the application to stop responding to user input (i.e. the protected application does not respond to mouse clicks) if the correct HASP HL is not attached, and to resume normal operations only if the key is re-connected. Only the functioning of the protected application is affected. Other programs should continue to function.

When manipulating code as described above, remember to account for legitimate users who have inadvertently forgotten to connect the HASP HL.

Use Data Stored in HASP HL Memory

When using data stored in the HASP HL memory, you typically check that the value is valid before proceeding. However, the verification process forces you to include the real value that is checked in your application. The value is then vulnerable to potential attacks.

Do not directly verify the value of the data. Instead you should do the following:

1. Read the data stored in the HASP HL memory.
2. Checksum the data using the guidelines specified in "Checksum code" on page 94.
3. Verify the checksum.

This procedure enables you to access data stored in the HASP HL memory without explicitly having to check its validity. If the required HASP HL is connected, the checksum value should be valid. Otherwise, the value is determined to be invalid and results in an error.

HASP HL Envelope - Best Usage

This section includes tips for best usage of HASP HL Envelope. Although the tool offers a wide range of variable protection parameters, the actual protection implementation is straightforward.

When protecting a series of executables, the HASP HL Envelope, with its multi-layer wrapping technology, implements a different protection scheme for every executable file. Varying protection schemes makes it much more difficult to understand and crack an application.

HASP HL Envelope Protection Trade-offs

When using HASP HL Envelope you have to decide between the level of protection and the following:

- Size of protected applications
- Necessary launch time for protected applications

When the protection level is set high, file size increases and the protected application takes longer to launch.

If the file size or launch time of the protected application are of no concern, the highest protection levels in HASP HL Envelope can be set accordingly.

The default settings in HASP HL Envelope offer an optimal balance between security and protected application considerations. HASP HL Envelope adds an effective protective shield to the application, while optimizing protected file size and launch time.

Enable background checks

Make sure you activate the background check setting. If a user disconnects the HASP HL in order to load the protected software on another terminal, the protected software instructs the user to reconnect the key; otherwise the software ceases to function. For more information see "Protection Attributes" (page 77) or the HASP HL Envelope online Help.

Update HASP HL Software

You should periodically update the HASP HL protection software. Updates are posted on the Aladdin corporate web site.

Access the Aladdin web site using the link provided in the Vendor Center suite to download the latest versions of HASP HL software.

Aladdin Knowledge Systems invests a lot of effort in maintaining its software protection technology and ensuring that it remains at the cutting-edge of the software-protection industry. HASP HL software updates are released regularly to deal with evolving software protection issues.

To adequately protect your system, always use the most up to date technology.

Chapter 8
HASP HL Licensing Overview

This chapter is an overview and an introduction to HASP HL licensing. It includes the following topics:

- Key HASP HL licensing concepts
- Available HASP HL licensing hardware and software
- Planning your HASP HL licensing schemes
- Working with HASP HL licensing

The following two chapters discuss in detail the software used for HASP HL licensing.

For information on HASP HL Factory, see the chapter "HASP HL Factory Licensing" (page 109).

For information on the Remote Update System, see the chapter "Remote Update System" (page 127).

Key HASP HL Licensing Concepts

In order to facilitate your use of the HASP HL system for your licensing requirements, it would be useful for you to understand the system's basic licensing concepts. Once you understand the concepts, you can evaluate and plan how to best use the HASP HL system to meet your particular licensing needs.

The following concepts are covered in this section:

- Protect Once-Deliver Many™
- Feature
- License
- Update
- Concurrence

Protect Once-Deliver Many™

This concept was first introduced in the HASP HL concepts chapter — refer to "Protect Once-Deliver Many™" (page 13). From a purely licensing perspective, it is important to reiterate the points below:

a. In the HASP HL system, licensing and protection are treated as two distinct and separate processes. You protect your software using one or both HASP HL protection methods:

- HASP HL API, refer to "HASP HL API Protection" (page 53).
- HASP HL Envelope, refer to "HASP HL Envelope Protection" (page 69).

HASP HL licensing does not concern itself with how you have applied protection to your software.

b. Licensing is an ongoing process that continues well after a program is protected. Whereas protection only concerns itself with security issues and is implemented once, the HASP HL system facilitates a licensing process that evolves and changes over time. Therefore, when introducing or modifying a licensing scheme, you do not have to re-protect your software and use up valuable development and testing resources. The system provides you with the necessary tools to enable you to modify and update licenses for your protected software.

Features

In the HASP HL system, a feature can be an entire application, an executable file or a specific functionality such as “Print”, “Save” or “Draw.” Features are the parts of your software you want to independently control in terms of user access and distribution.

Licenses

Licenses are a combination of the terms you apply to a feature, as well as the defined feature itself within a HASP HL-protected application. Licenses are created using HASP HL software and stored inside a HASP HL key.

Updates

Updates are modifications applied to licenses stored inside a HASP HL key. Updates can overwrite or supplement existing licensing information. In HASP HL, you do not need to recall your deployed keys to update licenses, you can apply the updates remotely.

Concurrence

Concurrence is a specific licensing term applicable to licenses in a network environment. Concurrence relates to the simultaneous access and usage of licensed features. It is controlled via a specific HASP HL utility — the HASP License Manager — which is deployed at your customer’s site. For more information, refer to "HASP License Manager" (page 151).

Available HASP HL Licensing Technology

This section surveys and lists all the software and hardware that is available in the HASP HL system for licensing your protected software.

Hardware

Table 8.1 lists the available HASP HL models that can be used for licensing, and the specific licensing terms associated with each model.

Table 8.1 HASP HL Hardware Licensing Capability

Hardware Model	Max Licenses	Highlights
HASP HL Pro	16	Activations Memory
HASP HL Max	112	Activations Memory
HASP HL Time	112	Activations Memory Real-time clock
HASP HL Net	112	Activations Memory Network environment

Software

The HASP HL system provides you with cutting-edge software for creating and managing licenses for your protected applications. The following software is used for licensing:

- **HASP HL Factory** — is part of the Vendor Center program suite. Use this tool to define and apply licenses for your protected software. Refer to "HASP HL Factory Licensing" (page 109) for more information.
- **HASP HL RUS** — this utility is deployed at your end user's site and is used to update HASP HL licences. Refer to "Remote Update System" (page 127) for more information.
- **HASP License Manager** (LM) — only used in conjunction with HASP HL Net keys. This application is distributed to end users. For more information, refer to "Operating the HASP License Manager" (page 154).

Planning for HASP HL Licensing

In order to get the most out the HASP HL's licensing capabilities, some advance planning is highly recommended.

HASP HL Factory enables you to define licensing elements and use these elements to set up your licensing schemes. You should consider the following issues:

1. What features within the protected software should be licensed?
2. How should these features be licensed?
3. How can these features be best combined to facilitate sales or business models?

Derive Features from Software

The first step in evaluating and planning your licensing requirements involves understanding the functional components that make up your software. Most software can be broken down into multiple functional components.

Breaking down your programs into features is an essential part of preparing a licensing scheme. We recommend that you list all the features you would like to license.

For example, a fictitious company providing design software for the construction industry identifies three specific features which it wants to license:

- Printing reports - PRINT REPORT
- Saving projects - SAVE
- Generating project reports - REPORT GENERATOR

The features PRINT REPORT, SAVE and REPORT GENERATOR can each be assigned a specific number as shown in Table 8.2.

Table 8.2 Sample feature list

Feature list	
Feature	**Assigned number**
PRINT REPORT	34
SAVE	4
REPORT GENERATOR	48

Use HASP HL Factory to incorporate the above list. For more information, refer to "Assigning identifiers to features" (page 113).

Combine Features to form “Packages”

Once you have identified features to license, you can create packages that are essentially combinations of these features.

HASP HL Factory enables you to design packages containing the features you have defined in the system. For more information, refer to "Packages" (page 114).

For example, you could design a “Trial” package that is earmarked for customers evaluating your software. In this case, you would only include a minimal combination of features. You could also have an “Enterprise” package that includes all the features and is targeted at large corporate customers. In between a “Single user” package is aimed at small-scale customer. This package offers more features than those available in the trial package but less than what is offered in the enterprise package.

In HASP HL Factory you define packages, but it is up to you to decide what kind of packages you want to create and their contents.

Table 8.3 Sample Packages and their contents

Feature	Trial	Single User	Enterprise
PRINT REPORT	**Included** Terms: 5 activations per license	**Included** Terms: 500 activations per license (trial)	**Included** Terms: Unlimited use
SAVE	**Included** Terms: 5 activation per license	**Included** Terms: unlimited usage	**Included** Terms: Unlimited use
REPORT GENERATOR	Not included	**Included** Terms: Subject to expiry date	**Included** Terms: up to 250 concurrent stations

Table 8.3 summarizes the three package options for the sample software company. The REPORT GENERATOR feature is not available in the trial package. The “single user” and “enterprise” packages contain the same features but have different licensing terms applied to each licensed feature.

Licensing Features

HASP HL enables you to control features through License Forms. These forms apply certain terms that independently control the use of the feature in question. You can control feature usage through a license by:

- Setting an activation counter for a feature
- Setting an expiry date for a feature
- Controlling concurrent usage of a feature

For each type of feature control, HASP HL Factory provides a particular License Form.

In the example shown in Table 8.3, the imaginary software firm has determined how it wants to condition the use of the features it has included in several packages. Although the Single User and Enterprise packages include the same features per se, they however condition feature usage differently.

The company would have to distribute HASP HL Net with its Enterprise package to control concurrent usage of the REPORT GENERATOR feature, while HASP HL Time would be distributed with the Single User package because only this HASP HL model contains a real-time clock to monitor expiry dates.

The choice of HASP HL models distributed together with your protected software should reflect the way you would like to control the use or access to a licensed feature.

HASP HL Licensing in Action

This section provides an overview of how HASP HL licensing tools are used to implement and execute licensing schemes.

Stage 1. Use HASP HL Factory to define elements

Once you have planned licensing schemes, use HASP HL Factory to define your features, licensing terms and packages. The tool also helps you determine which HASP HL model listed in Table 8.1 (page 102) is most suitable for your licensing schemes.

Once you have defined features within HASP HL Factory, export the list of features to your software developers. The list will serve as a reference when implementing the specified identifiers for each feature. For more information on the protection side of the feature identifiers, refer to "The HASP HL API Login Function" (page 61).

Stage 2. Use HASP HL Factory to set up orders

Use the licensing elements defined in stage 1 to create orders for you customers. Orders in HASP HL Factory apply licensing terms to a HASP HL key. For more information, refer to "Setting up an order" (page 119).

Stage 3. Use HASP HL Factory to execute orders

The licensing data contained in the order is stored in the key. For more details, refer to "Executing Orders" (page 122).

Stage 4. Distribute Software and keys

Distribute the keys containing the licences to your customers together with your protected software. Make sure that you provide your customers with the required drivers to enable the HASP HL keys to run on their systems. For more information, refer to "Distributing HASP HL Drivers and Daemons" (page 139). You should also distribute the HASP License Manager to customers to whom you have supplied HASP HL Net keys. For more details, refer to "HASP License Manager" (page 151).

Stage 5. Update your deployed licenses

HASP HL enables you to update licenses that are deployed at your customer’s site. For more details, refer to "Remote Update System" (page 127). Use the HASP HL RUS utility to get information on deployed HASP HL keys and to supply updates. The updates are created using HASP HL Factory. For more information, refer to "Executing Orders Through the Remote Update System" (page 123).

Chapter 9
HASP HL Factory Licensing

This chapter describes how HASP HL Factory is used to create, manage and implement your licensing schemes. It includes the following topics:

- Prerequisites for using HASP HL Factory
- HASP HL Factory data structures
- Creating HASP HL Factory orders
- Executing HASP HL Factory orders

Introduction

HASP HL Factory is one of the three programs included in the Vendor Center suite of programs. Its main purpose is to enable you to define and create licensing data which is stored in the HASP HL key. HASP HL Factory is also used to directly apply data to the memory of a HASP HL key or to produce an update file which is remotely applied to a deployed key.

Unlike HASP HL tools used to apply protection to your software, (see "HASP HL Envelope Protection" (page 69) or "HASP HL API Protection" (page 53)), HASP HL Factory does not interact with or require any elements of your software.

As a licensing tool, HASP HL Factory is used to define licensing elements for your protected applications, and to apply licenses in HASP HL keys. HASP HL Factory is also used to write specific data into the memory of a HASP HL key.

HASP HL Factory enables you to view licenses stored in HASP HL keys either directly, if the key is available and connected to your machine, or remotely using the Remote Update System.

A **Sample** is included with your version of HASP HL Factory to help you understand the data structures for the various licensing elements. HASP HL Factory also includes an Online Help.

Prerequisites for HASP HL Factory

There are three levels of prerequisites for successful use of HASP HL Factory:

- Software
- Hardware
- Advance planning of licensing schemes

Software requirements

You should have all of the following installed on your system:

- HASP HL drivers or daemons.
- HASP HL software including the Vendor Center suite of programs.
- A valid Vendor Code stored in the *Vendorcodes* folder. Refer to "Extracting the Vendor Code from the Master HASP HL" (page 56) for more details. The DEMOMA Vendor Code is always available when using the HASP HL Demo key provided with the HASP HL Developer's Kit.
- One of the following Windows operating systems: Windows 2000/XP or any newer Windows platform.

Hardware requirements

If you want to use HASP HL Factory to execute orders, you need:

- A Master HASP HL with a specific Batch Code. You can introduce new Master HASP HL keys through HASP HL Factory.

If you are evaluating HASP HL Factory, use the provided **Sample** to execute orders. The Sample is an integrated batch and does not require a Master HASP HL.

- A HASP HL into which you will store the order. This key must meet your specific licensing requirements. For information on the licensing capabilities of each HASP HL model, refer to "HASP HL Hardware Licensing Capability" (page 102).

Planning requirements

As mentioned in the previous chapter, it is recommended that before you use HASP HL Factory, you evaluate and plan your licensing requirements. For more information, refer to "Planning for HASP HL Licensing" (page 103).

After evaluating and deciding on your licensing requirements, use the sample provided to study HASP HL Factory data structures.

HASP HL Factory Data Structures

The following elements form the basic data structures used in HASP HL Factory:

- Batches
- Features
- Packages
- Orders

Batches

When you order HASP HL keys, your company is assigned a Batch. This assigned batch is used to process orders from you company. All keys produced for a batch are subject to a unique and specific encoding. Therefore keys belonging to a particular batch have a unique encryption and decryption behavior.

Batches are used in HASP HL Factory to define features and packages, and to execute orders. Typically a single batch is required to license all your products. However, a new batch entry is added to HASP HL Factory whenever a Master HASP HL containing new batch data is introduced to the system. This data includes:

- A Batch Code — A five to eight character code assigned uniquely to every HASP HL customer, and used specifically to order keys. The code is printed on the HASP HL label of every key.
- A Vendor Code —This unique code is required by the HASP HL system when protecting software or creating licenses. Refer to "Extracting Vendor Codes" (page 44) for more information.

Master HASP HL keys with new batch-related data are detected on start up of HASP HL Factory. At any time when using the tool, you can ask the system to introduce a new Master HASP HL after connecting the key with the new batch data.

Once a batch is introduced to HASP HL Factory, you can define features and packages belonging to this batch.

Features

Together with packages, features are licensing elements used to facilitate the process of producing licenses for protected applications. HASP HL Factory enables you to define and control the use of features included in your software.

Features are displayed as a part of every batch entry in HASP HL Factory.

Assigning identifiers to features

The main task in defining a feature in HASP HL Factory is to assign a unique numerical identifier to the feature and the name of the feature. Once stored in the system, the feature is thereafter represented by the assigned name and number. HASP HL Factory ensures that no two features share the same numerical identifier. Usually the assigned number remains unchanged over the life span of the product

Available identifiers

Table 9.1 below lists the available program number range for every HASP HL model. The number zero is reserved for the default feature, which is a transparent feature that is always available whenever the required HASP HL is connected. The default feature does not have to be configured by HASP HL Factory and takes upon settings determined by the connected HASP HL.

Table 9.1 Available Program Numbers for HASP HL Models

HASP HL Model	Program Numbers
HASP HL Basic	n/a
HASP HL Pro	1 to 16
HASP HL Max	1 to 112
HASP HL Net	1 to 112
HASP HL Time	1 to 112

Once you have introduced features into HASP HL Factory, you can use these features to create packages or orders.

Packages

A package is any combination of the following elements:

- Features
- Licensing terms of the included features
- Data that is written to the memory of a HASP HL key

Packages are defined in the HASP HL Factory application to facilitate the process of licensing protected applications. Packages are displayed as a part of every batch entry in HASP HL Factory.

Creating packages

To create a package in HASP HL Factory, you first need to assign a package name. You can then proceed to do any of the following:

- Include a defined feature in the package. This feature becomes part of the package properties.

- Define input that is to be written to the memory of a HASP HL key. The memory input becomes part of the package properties.

Packages are valid if they include one or both of the properties described above.

Feature licensing terms

When adding a feature to a package, you need to define licensing terms for the feature. Your licensing plans (see "Planning for HASP HL Licensing" on page 103) determine the selection of licensing terms.

License Forms

Licensing terms are applied through License Forms in HASP HL Factory. There are three types of HASP License Forms available in HASP HL Factory:

- **Counter** — used to control the number of activations for a feature.
- **Timer** — used to control the expiry date for a feature.
- **Net** — used to control concurrent usage for a feature.

Regardless of the type of form you select, the actual licensing terms are defined in three possible ways:

- Use **Set** when the you want the licensing terms you define to overwrite whatever licensing data exists in the HASP HL key. You would usually use the **Set** option for packages in which you want the licensing terms to reflect a definite value.
- Use **Add** when the you want the licensing terms you define to supplement existing data in the HASP HL key. You would usually use the **Add** option for packages in which you want the licensing terms to add and not overwrite data in the key.

- Use **Remove** to terminate a particular licensing term for a feature in a HASP HL key. You would usually use the **Remove** option to disable a particular licensed feature.

In the HASP HL licensing system, a **perpetual license** is represented by **unlimited counter.**

Package memory inputs

Use the memory content editor included in HASP HL Factory to set data that is to be written to the memory of a HASP HL key. This data can be defined as part of a package. The input for the memory can target a specific position, therefore you can have multiple chunks of memory input written at different locations. The HASP HL memory can therefore accommodate multiple inputs provided by various packages.

You can enter data in binary form or as ASCII characters. Data input to the memory can be use to:

- Assign a unique code to each software user.
- Save passwords, program code, program variables, and any other data.

Use multiple packages to built a combination of memory chunks in the memory of a deployed HASP HL key.

Orders

An order is licensing data that is transferred to a HASP HL key. Licensing data includes definitions of how protected features are controlled and modifications applied to the memory of a key. When an order is applied for the first time into a key, the key is initialized and activates the licensing scheme specified in the order.

Orders in HASP HL Factory have three possible forms:

- Orders that can be applied to any HASP HL that meets your licensing requirements. These orders are applied to ‘empty” license containers within the HASP HL keys. In this case, all the data previously stored in the keys will be lost.
- Orders that contain updates for specific HASP HL keys. If the keys are available, you can use HASP HL Factory to apply the updates directly to the keys to modify the data stored in the keys.
- Orders that contain updates for specific HASP HL keys which are currently deployed in the field and not available. These orders are applied using the Remote Update System.

You can use HASP HL Factory to execute multiple orders. Simply select all the queued orders you want to execute.

HASP HL Factory queues and displays all the orders that have yet to be executed. This facilitates the separation of order processing from order production.

Orders are tracked in the internal HASP HL Factory database and are no longer displayed after they have been executed.

Creating HASP HL Factory Orders

Prerequisites

To create an order you need the following:

- Master HASP HL. For evaluation purposes you can use the **Sample** included in HASP HL Factory to create and execute orders.
- Your own batch entry defined in HASP HL Factory. For evaluation purposes, you can use the included **Sample.**
- A HASP HL into which you execute the order.
- Optional: features and packages defined in HASP HL Factory.

To facilitate the order creation process in HASP HL Factory, we recommend that you define all your licensing elements (packages and features) beforehand.

Stages

Order management in HASP HL Factory involves a two-phase process.

1. Setting up the order.
2. Executing the order.

The two-phase process facilitates the separation of responsibilities between order processing and order production.

Setting up an order

To set up orders in HASP HL Factory, your orders must have the following properties.

Batch

This is the name of the batch creating the order. All packages and features defined for this batch can be used to create the order. For more information, refer to "Batches" (page 112).

Name

An identifier used to recognize the order. Typically this would be the name of the order recipient.

Keys containing licensing data are recognized by the system and their names displayed when connected to the computer running HASP HL Factory.

Comment

This is an optional property containing a description related to the order.

Order types

You can use HASP HL Factory to create three kinds of orders.

a. Generic orders. These orders are contain the initial licenses that are stored in a key before it is shipped to customers together with the protected software. This type of order initializes the licensing and memory of the HASP HL key.

b. Updates to connected keys. Once a key is initialized, its memory contents and licensing can be modified when the is connected to PC running HASP HL Factory. You can add, reset or delete the current licensing status of the key.

For more information, refer to "Executing an Order to a Connected HASP HL key" (page 122).

c. Updates which are applied remotely to deployed HASP HL key. HASP HL Factory produces the license update and the end user has to apply the new data. For more information, refer to "Applying an update" (page 133).

Adding a Feature

You can only add features to orders if the features are already defined in HASP HL Factory.

Depending upon your licensing plans, when adding a feature to an order, you need to define licensing terms for the feature. For more information, refer to "Planning for HASP HL Licensing" (page 103).

License Forms

Licensing terms are applied using License Forms in HASP HL Factory. There are three types of License Forms available in HASP HL Factory.

- **Counter** — used to control the number of activations for a feature.
- **Timer** — used to control the expiry date for a feature.
- **Net** — used to control concurrent usage for a feature.

Regardless of the type of form you select, the actual licensing terms are defined in three possible ways:

- Use **Set** when the you want the licensing terms you define to overwrite whatever licensing data exists in the HASP HL key. You usually use the **Set** option for orders in which you want the licensing terms to reflect a definite value.
- Use **Add** when the you want the defined licensing terms to supplement existing data in the HASP HL key. You would usually use the **Add** option for orders in which you

want the licensing terms to add to, and not overwrite, data in the key.

- Use **Remove** to terminate a particular licensing term for a feature in a HASP HL key. You would usually use the **Remove** option to disable a particular licensed feature.

Adding a Package

You can only add packages defined in HASP HL Factory for the batch entry under which you are creating the order. Once the package is included in the order, all its contents, including features, licensing terms and input for the HASP HL memory, become part of the order. You can edit all the data included in the order.

Order Memory Input

Use the memory content editor included in HASP HL Factory to set data that is to be written to the memory of a HASP HL. This data can be defined as part of an order. You can enter data in binary form or as ASCII characters. Data input to the memory can be used to:

- Assign a unique code to each software user.
- Save passwords, program code, program variables, and any other data.

The memory content editor reflects the entire contents of the HASP HL memory. This is not the case with packages. The editor in this case only reflects a section of the memory that is modified when the package definition is applied to a key.

Executing Orders

There are three ways to use HASP HL Factory to ‘target’ your orders:

- You an execute an order that initializes the HASP HL key for the first time, usually a key that has not been deployed at an end user’s site. This order is not an update for an existing license.
- You can execute an order directly to a HASP HL if it is available and connected to the computer on which HASP HL Factory is running. This order is an update for an existing license.
- You can create an update which is applied remotely using the HASP HL RUS utility. This order is an update for an existing license.

Executing an Order to a Connected HASP HL key

Once your order is queued it can be applied to a connected HASP HL. You can only modify an order before it is executed. Once the order is executed, the order is tracked by the internal database of HASP HL Factory, but it is no longer displayed.

Make sure that the key to which you apply the order meets your licensing requirements. For example, if your order contains a feature licensed using an expiry date, you should apply the order to a HASP HL Time.

Procedure

Once an order is set up, it can be executed. To execute an order you need the following:

- Master HASP HL for the Batch under which the order was created.
- A HASP HL — this key stores the order.

To execute an order:

1. Connect the Master HASP HL for the batch under which the order was created.
2. Connect the HASP HL into which the order is being executed.

To apply an order that contains an update for a specific key, you need to connect the specific key. To apply an order to an 'empty' key, you can connect any HASP HL as long as the model adheres to the license tasks and requirements stipulated in the order.

3. Select the order(s) you want to execute from the list displayed in HASP HL Factory.
4. Execute the order(s).

Executing Orders Through the Remote Update System

If you are updating a license deployed in the field, use HASP HL Factory to produce the license update data. You can use HASP HL Factory to produce updated information in two formats used by the remote update system:

- As an executable which will include *hasprus.exe*. Your customers simply need to run the executable.
- As a V2C file. For more information on using the this format, refer to "The HASP HL RUS Utility" (page 131).

For more information on how the HASP HL system handles remote updates, refer to "Remote Update System" (page 127).

Viewing HASP HL Keys

HASP HL Factory enables you to view the status of a HASP HL key after an order has been applied to it. The key must be connected in order to view its features and the contents of its memory.

To view the contents of a HASP HL key:

1. Connect the key you want to view.
2. Select **View Connected Key** from the **Tools** menu.
3. The **Key Properties** dialog contains the following information:
   - Batch — the identifier of the batch used to create the license data in the key.
   - Name — the identifier for most recent order applied to the key.
   - HASP ID — the unique serial number for the key.
   - Hardware — the HASP HL model.
   - Features — click on the Features tab to view the list of included features.
   - Memory — click on the Memory tab to view the memory image for the key.

You can also view the contents of a deployed key by using a C2V file containing information on the license status of a HASP HL. For more information, refer to "Using HASP HL RUS" (page 132).

Creating an update for a specific key

You can use HASP HL Factory to create and apply an update directly to a connected HASP HL.

1. View the Key properties.
2. Click **Create Update**.

Creating an update essentially creates a new order for a specific key. The order automatically loads all the current contents of the key. You can choose to either modify the data or add new data to the key.

If you delete a listed feature from the key, this feature will not be affected by the update. All features in the Order Editor override existing features in the key when the order/update is executed. To remove a license for a feature you must use the Remove option made available in the HASP License Form.

Once the order is queued and ready you can use HASP HL Factory to apply the order directly to the key. You can also generate a V2C file containing updated licensing information which is shipped to the end user.

Reading Time in HASP HL Keys

Before distributing HASP HL keys with licensing data, you can use HASP HL Factory to read and set the real-time clock (RTC) in HASP HL Time keys. For more information, refer to the HASP HL Factory online Help system.

The RTC is preset to UTC.

Chapter 10
Remote Update System

This chapter describes the HASP HL Remote Update System (RUS). It includes the following topics:

- Components of HASP HL RUS
- How the HASP HL RUS works
- Using the hasprus.exe utility
- Configuring V2C output

Concept

The HASP HL RUS is an advanced mechanism that enables secure, remote updating of deployed HASP HL keys. The mechanism enables you to update the license and memory of a HASP HL that is in your customer's possession. You can update the following HASP HL models using the system: HASP HL Pro, HASP HL Max, HASP HL Net and HASP HL Time.

HASP HL RUS is a simple and secure way to remotely manage your licenses even after you have delivered your protected software together with the HASP HL keys. You simply need to update the license and deliver update files to your customer, for example by e-mail. HASP HL RUS also enables you to get information on the current status of licenses at your customer's site. The system even provides a secure mechanism to reduce license functionality at your customer's site.

As part of the basic concept underlaying the HASP HL system, the Remote Update System facilitates ongoing licensing well after protection has been implemented. For more information, refer to "Protect Once-Deliver Many™" (page 13).

By using the HASP HL RUS, you do not need to recall keys from your customers whenever you want to modify the terms of a licensed feature.

Components

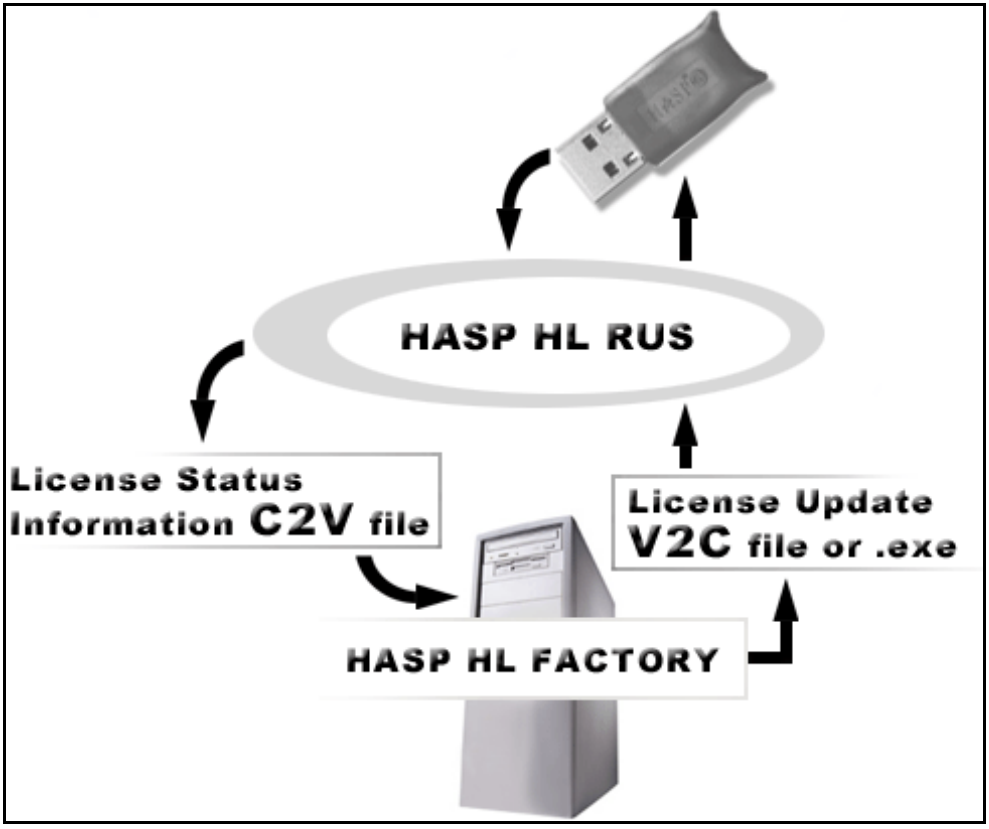
The HASP HL RUS is composed of the following components:

- **HASP HL Factory** — this vendor tool is used to create the updates that are delivered to end users. It is also used to read the files sent by end users on the current licensing status of the keys in their possession. Update information is conveniently created in the same tool used to generate the original licensing information. Updates are handled in HASP HL Factory as a particular order type. For more information on how orders are created, refer to "Creating HASP HL Factory Orders" (page 118).
- **HASP HL RUS** — is the name of the updating utility that is distributed to end users. The utility — *hasprus.exe* — is used to collect information on the licensing status of keys deployed in the end-user's site. It is also used to apply updates to licenses. This utility can be customized and is generated by HASP HL Factory.
- **C2V files** — customer to vendor files contain licensing data sent to you by your customers. These files have the *.c2v* extension and contain licensing information in deployed keys. To produce a C2V file, your customers have to click a button in the *hasprus.exe* and save the information. Once you receive a C2V file, you can view its contents using HASP HL Factory. In addition to licensing information, the C2V files contain an image of the contents of the deployed HASP HL memory.

- **V2C files** — vendor to customer files are the actual license updates for deployed HASP HLs. You produce these files using HASP HL Factory. HASP HL Factory enables you to chose either to send the V2C files as self-executing entities or as files with the *.v2c* extension that your customers apply via the HASP HL RUS utility. In addition to licensing information, you can include data to be written to the memory of a deployed HASP HL.

The relationship between the various components that makeup the HASP HL RUS system is summarized in Figure 10.1.

Figure 10.1 HASP HL RUS Components

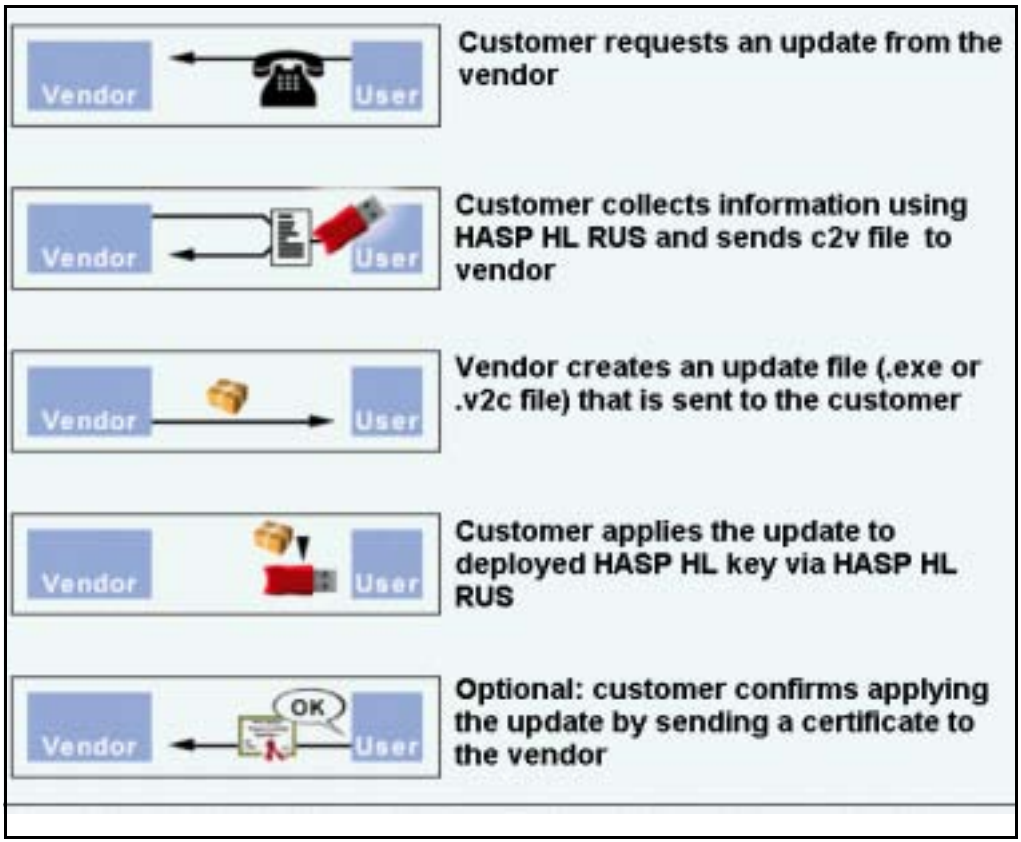


Remote Update System Workflow

The tools and utilities described in the previous section support a secure and simple workflow. Figure 10.2 presents a sample workflow.

A customer wants to update a licensed feature and contacts the software vendor. Before an update can be produced, the vendor needs to receive the C2V file for the deployed key. This file contains the current status of the license at the customer's site, as well as the contents of the memory of the HASP HL.

Figure 10.2 Sample HASP HL RUS Workflow



The collected data enables the vendor to produce an update most suited to the customer's needs. At no point in this workflow is it necessary to reconfigure security or protection at the end user site. The vendor produces an update (or V2C file or executable) that is shipped to the end user. The end user applies the update.

The vendor can specify in the v2c file whether or not a update confirmation from the end user is required. This confirmation is produced at the end user site once the update is applied. The end user needs to send the confirmation to the vendor.

The HASP HL RUS Utility

The HASP HL RUS utility is the HASP HL software at the end-user site responsible for:

- Collecting the current licensing data (including memory content) in deployed keys.
- Applying updates sent to the customer by the vendor.

Use the HASP HL Factory tool to produce and customize the utility for your customers.

You can opt to integrate a copy of the utility together with the other HASP HL software distributed to your end users. For more information on how and what HASP HL software is distributed to end users, refer to "Distributing HASP HL with your Software" (page 137).

You can also distribute HASP HL RUS separately after creating an update for a licensed feature within your protected software. The HASP HL Factory tool has an option to include the V2C file within an executable. For detailed information on all the various configuration possibilities for producing the HASP HL RUS utility using HASP HL Factory, refer to the HASP HL Factory online Help system.

The HASP HL RUS utility has a graphic user interface (GUI) that is divided into two tabs.

- **Collect Key Status Information** — tab is used to collect information on the current status of the licenses in the HASP HL key. The user needs to specify a name and location for the generated C2V file.
- **Apply License Update** — tab is used to apply the V2C file and update the licenses in the HASP HL.

When generating V2C information using HASP HL Factory, you can opt to customize the output to include both or only on of the tabs described above.

Using HASP HL RUS

The HASP HL RUS utility is designed with your customer in mind. Implementing either the license data collection or license update functionality is a straightforward and easy process.

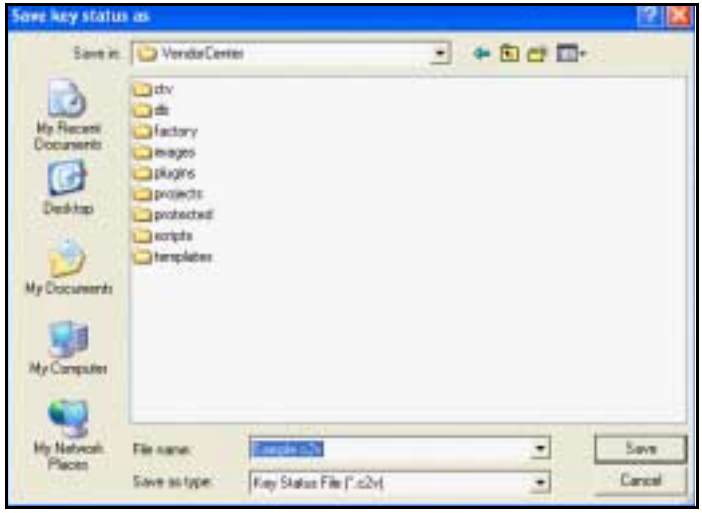
Collecting HASP HL license data

You need to customize HASP HL RUS using the HASP HL Factory to include all the information necessary for searching and detecting your deployed HASP HL.

To collect information on the current license status in a HASP HL key, follow the procedure outlined below.

1. Launch *hasprus.exe.*
2. Select the **Collect Key Status Information** tab.
3. Click Collect information. The **Save key status as** window shown in Figure 10.3 (page 133) is displayed.

Figure 10.3 Save key status as screen



4. Point to the directory where you want to store the C2V file. Enter a file name and click **Save**.

5. You have created a C2V file for the HASP HL key. The file can now be sent for processing to produce an update.

Applying an update

To apply an update to the licenses stored in a HASP HL key, follow the procedure outlined on the next page.

1. Launch *hasprus.exe* or click on the V2C file containing the update data.

If you have received an update as an executable, click on the file and it will automatically launch HASP HL RUS.

2. Select the **Apply License Update** tab. Only this tab might be displayed if the update is in an executable file.
3. If the **Update file** field is empty, browse to the directory where the update file (V2C file) is located.
4. Click Apply update to apply the new license data to the deployed HASP HL key.
5. You may be prompted that a receipt has been produced to confirm the update. Send this receipt for processing.

Branding V2C Output

When using HASP HL Factory to produce an update file to be processed at the end user's site, you can brand the V2C output.

An HTML editor is provided in the HASP HL Factory tool to brand instructions or messages for your end users. You can use this functionality to identify the source and contents of the update.

You can brand the following using the HASP HL Factory:

- A V2C file — this file will be uploaded by the end user using the HASP HL RUS you have provided them.

We recommend that you distribute your protected software together with a branded version of HASP HL RUS. The latter should be branded to the .v2c extension.

- An executable file — this file is an update that is attached to a HASP HL RUS utility. End users have to launch the executable and follow the on-screen instructions.

For more detailed information on customizing V2C output, refer to the HASP HL Factory online Help.

Chapter 11
Distributing HASP HL with your Software

This chapter introduces ways and means of distributing required software to your end users. It includes the following topics:

- What HASP HL software components should be distributed to end users
- Options for distributing HASP HL drivers for Windows platforms
- Options and means for distributing the HASP HL daemon for Mac and Linux platforms

HASP HL Software for End Users

Every HASP HL installation includes software that you should distribute to your end users. This software has to be installed at your customer's site to ensure that your protected and licensed software function properly.

You do not need to distribute all the software covered in this chapter.

The HASP HL drivers and daemons are the important software to deliver to end users.

Protection related software

The most important software that should be installed at your customer's site are the HASP HL drivers and daemons. These applications are required to enable the HASP HL keys to run and communicate with the protected application. For more information, refer to "Distributing HASP HL Drivers and Daemons" (page 139).

Network environment software

If you are shipping HASP HL Net keys to end users, you will need to also deliver the following applications:

- **HASP License Manager** — to control usage of your protected and licensed software within a network. For more information, refer to "HASP License Manager" (page 151).
- **Aladdin Monitor** — used together with the HASP License Manager and HASP HL Net keys. For more information refer, to "Aladdin Monitor" (page 181).

Troubleshooting software

You should consider distributing Aladdin DiagnostiX to all your end users. This utility is used to collect information on the deployed HASP HL key and the environment in which it operates. For more information, refer to "Diagnosing HASP HL Keys" (page 189).

Software to update licenses

The HASP HL RUS utility is distributed when remotely updating licenses in deployed HASP HL keys. For information on this utility, refer to "The HASP HL RUS Utility" (page 131).

Distributing HASP HL Drivers and Daemons

The drivers and daemons for the HASP HL keys in your customer's possession must be installed on their systems. Without the required drivers or daemons, your protected programs will not run because they are unable to communicate with the HASP HL.

The following sections describe the various options available for distributing HASP HL drivers and daemons to your end users.

Distributing HASP HL Drivers for Windows

HASP HL device driver distribution options are listed below.

- Using Windows Update or Aladdin DiagnostiX to download HASP HL drivers. Internet connection is required for both these approaches.
- Integrating the installation of the HASP HL driver installation into your application's installer using the two options below:
  - Merge modules
  - HASP HL Driver Installation API
- Delivering to your end users either of the following HASP HL Device Driver installation utilities:
  - *HASPUserSetup.exe* — a GUI-based installer
  - *haspdinst.exe* — a command line utility

Windows Update

If your end users are running the protected software on Windows XP or 2003 platforms, and can access the Internet, they simply have to connect the HASP HL keys to their machines. All HASP HL drivers are certified by Microsoft and are therefore automatically obtained and downloaded from the Microsoft Update site.

Once your end users connect the HASP HL keys:

1. The system informs them that new hardware is detected.
2. The drivers for the HASP HL key are automatically installed.
3. When the LED on the HASP HL key lights up, the installation process is complete.

Another way to confirm whether or not the required drivers have been installed is through the Device Manager utility available on all Windows platforms. The Aladdin device drivers for every connected HASP HL keys should be listed under **Universal Serial Bus Controllers**.

Figure 11.1 Device Manager listing Aladdin device drivers

Using Aladdin DiagnostiX

Once your protected software and HASP HL keys are deployed at the end user's site they can use the **Update Driver** functionality offered in Aladdin DiagnostiX. Your end users need an internet connection and should install *askdiag32.exe* on the computers running the protected applications. For more information, refer to "Updating HASP HL Drivers" (page 196).

Unlike the Windows Update option described in the previous section, Aladdin DiagnostiX is also relevant to platforms older than Microsoft XP or 2003.

Merge Modules

You can use HASP HL merge modules to seamlessly integrate HASP HL driver installations within your MSI installation. Merge modules deliver shared Windows Installer® components and setup logic for an application. Shared code, files, resources, registry entries and setup logic are contained within a single composite file. The HASP HL driver installation is available as the *haspds.msm* merge module.

The *haspds.msm* merge module cannot be run as a stand alone application.

The *haspds.msm* merge module is located in the following directory on the HASP HL installation CD:

`Windows/Installed/API/DriverInstall/MSI`

A sample merge module is provided in the following directory:

`Windows/Installed/Samples/DriverInstall/MSI`

Concept

A *haspds.msm* integrated to your MSI installer copies the *haspds_windows.dll* into the Win32 system directory of the end user's PC. The *hasp_ds_windows.dll* is called by the MSI module to install or uninstall the HASP HL drivers.

The benefits of using the HASP HL installation merge modules within a single unified MSI installer include:

1. Providing end users with a single compound file for your application that includes the HASP HL installation.
2. Installation self repair provided by reusing the MSI installer.

Implementation Checklist

Before implementing HASP HL merge modules into your installer, please review the following checklist:

- End users will require "administrator" rights in order to successfully execute the driver installation. Please ensure that this is accounted for in your install scripts.
- HASP HL merge modules require Windows Installer version 2.0 or later.

Do not use earlier versions of Microsoft Installer!

- No open processes, requiring HASP HL drivers, should run in the background when installing drivers.
- Refer to the *haspds.msm* sample for MSI for a demonstration of HASP HL merge modules in action.

Implementation

Implementing the HASP HL merge modules is a straightforward process simply requiring you to add the **.msm* file containing the driver installation to your MSI-compliant installer setup. Once you have created your MSI installer, the wrapped file will automatically include the HASP HL installation merge module.

Before implementing the merge module, you should review the implementation checklist.

Sample merge module

A sample installer containing the HASP HL merge module is provided and should be studied before implementing the *haspds.msm* merge module into your own installer.

The sample installer is a full MSI-installer containing the HASP HL driver installation merge module and the required shared libraries for installing HASP HL drivers.

The sample installer does the following:

- Verifies that the user has the requisite administrator rights to install the HASP HL drivers.
- Stops a running HASP License Manager service before the driver is installed, and re-starts the service once the installation is complete. For more information on the HASP License Manager, refer to "Activating and Deactivating HASP License Manager" (page 154).
- Installs or removes the HASP HL drivers.

HASP HL Driver Install API

Use the HASP HL Driver Install API to integrate the installation process into your custom setup application. Refer to the separate help file in the directory listed below for more details

```
Program Files/Aladdin/HASP HL/Drivers/
```

A sample for the HASP HL Driver Install API is provided on HASP HL installation CD in the following directory:

`Windows/Installed/Samples/DriverInstall/C/`

Use the above sample and the Device Driver help files to learn how to use HASP HL Driver Install API for your setup applications.

haspdinst.exe

The *haspdinst.exe* utility is a command-line application that installs the HASP HL device drivers under Windows 98/ME/ 2000/XP/Server 2003 systems. You can distribute this stand-alone application to your end users. A copy of *haspdinst.exe* is

> ! You should only use this method if you are sure that your end users are capable and willing to use a command-line prompt.

available in the following directory on the HASP HL installation CD:

`Windows/Installed/Redistribute/Drivers/`

To install the HASP HL Device Drivers:

Type `haspdinst -i` in the command line.

For more information, refer to "The haspdinst.exe Utility" (page 35)

To confirm whether or not the required drivers have been installed, check for HASP HL device drivers entries in the Device Manager utility available on all Windows platforms. Entries should appear under **Universal Serial Bus Controllers** as shown in Figure 11.1.

HASPUserSetup.exe

A copy of *HASPUserSetup.exe* is available in your HASP HL installation CD in the following directory:

`Windows/Installed/Redistribute/Drivers`

This is an easy to use program that installs the HASP HL drivers under Windows 98SE/ME/2000/XP/Server 2003 systems. It has an intuitive GUI based wizard. Send this file to your end users. Once they launch the file, the Welcome screen shown in Figure 11.2 is displayed, they should follow the on-screen instructions until the driver installation is completed.

Figure 11.2 *HASPUserSetup.exe* Welcome Screen

To confirm whether or not the required drivers have been installed, check for HASP HL device drivers entries in the Device Manager utility available on all Windows platforms. Entries should appear under **Universal Serial Bus Controllers**.

Distributing HASP HL Daemons for Mac

You should distribute the HASP HL daemon — *aksusbd* — to end users running protected and licensed applications on Mac (OS X) platforms. Without the daemon, the end user's system will not be able to recognize the connected HASP HL keys, and the protected applications will not be able to run without access to the required keys.

All distributable HASP HL software for Mac is provided in the *MacOS/Redistribute* folder in your HASP HL installation CD. HASP HL software updates for Mac are also available at

`http://www.hasp.com/downloads`

Options for distributing Mac Daemon

All the software required to distribute the daemon is provided in the following directory on the HASP HL installation CD:

MacOS/Installed/Redistribute/Runtime/

This section presents a number of options for distributing the Mac daemon to your clients. Each method has its advantages. The methods are presented in descending order of preference.

Installation Image

An installation image — *HASP Installation.dmg* — is provided in the *dmg* sub-folder on the HASP HL installation CD. The image is not customizable but has the advantage of being compressed into a format that is easily downloaded. Another advantage is that the user privilege settings are set correctly.

Make *HASP Installation.dmg* available on your FTP site for your end users to download.

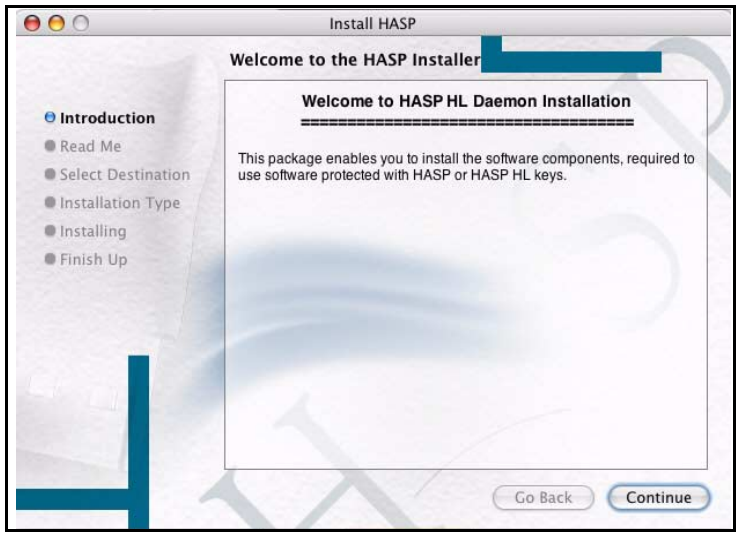
For instructions on how to use the *HASP Installation.dmg* to install the daemon, refer to *"Installing the daemon" (page 38)*.

Installation package

An installation package for the HASP HL daemon — *AKSUSB Install.pkg.sit* — is provided in the *pkg* sub-folder on the HASP HL installation CD.

1. Copy the contents of the *pkg* sub- folder to your local drive.
2. Use the **Stuffit Expander** utility to decompress *AKSUSB Install.pkg.sit*.
3. Run *AKSUSB Install.pkg* file.
   The window shown in Figure 11.3 is displayed. Follow the on-screen instructions.

Figure 11.3 HASP HL Daemon setup



The installation package can also be integrated into any multi-package installer. For more information, consult the relevant Apple Developer documentation at:

http://developer.apple.com

For more information on using the scripts, refer to the *readme.txt* file provided in the *pkg* sub-folder.

Installation package sources

The sources used to create the HASP HL daemon installation described in "Installation package" (page 147) are available in the *pkg-source* sub-folder. Table 11.1 lists the folder's contents.

Table 11.1 Mac OSX Daemon Installation source files

File	Description
aksusbd_install.pmsp	Project file for the daemon installer. Used by the PackageMaker utility when creating packages.
dinst_Contents.sit	A compressed file containing the *aksusbd* daemon, and all the start up parameters. The contents of this file can be configured to meet the needs of your clients before delivery.
dinst_Resources.sit	A compressed file containing background image for the installer and the shell scripts used to hand the pre and post installation processes.The contents of this file can be configured to meet the needs of your clients before delivery.
aksusbd_forceinstall.pmsp	Project file used for a forced installation in which all HASP HL daemon data is overwritten.
dinstforce_Resources.sit	A compressed file used for or a forced installation in which all HASP HL daemon data is overwritten. It does not contain shell scripts to handle the post and pre installation process.

To include the daemon source files into your installer, follow the procedure described below.

1. Copy the contents of the *pkg-sources* sub- folder to your local drive.
2. Use the **Stuffit Expander** utility to decompress all the compressed files listed in Table 11.1.

3. Integrate the files into your installer.
   For more information consult the relevant Apple Developer documentation at:

http://developer.apple.com

For more information on the installation sources, refer to the *readme.txt* provided in the p*kg-sources* sub- folder.

The relevant compressed files can be integrated into a single installer for your protected application. The end user automatically installs the daemons together with your protected and licensed software. You have the option of displaying the window shown in Figure 11.3 (page 147) when the end user runs your installer.

All the *aksusbd* files are automatically stored at the root volume at your customer's site. If you want to prevent this, we recommend that you use the installation package option.

Installation scripts

Installation scripts are provided in the *script* sub-folder on the HASP HL installation CD. Copy these scripts and send them to your end users who will use these scripts to install the daemon through a command-line shell. The scripts are not configurable.

You should only use this method if you are convinced that your end users are capable of using a command-line shell.

For more information on using the scripts, refer to the *readme.txt* file provided in the *Script* sub-folder.

Distributing HASP HL Daemon for Linux

You should distribute the HASP HL daemon — *aksusbd* — to end users that will run protected and licensed applications on Linux platforms. Without the daemon, the end user's system will not be able to recognize the connected HASP HL keys, and the protected applications will not be able to run without access to the required keys.

All distributable HASP HL software for Linux is provided in the following folder on the HASP HL installation CD:

Linux/Installed/Redistribute

HASP HL software updates for various Linux distributions are also available at

http://`www.hasp.com/downloads`

Available formats for HASP HL Daemon distribution

The *Linux/Redistribute* folder contains the HASP HL daemon for distribution in the following formats

- As an RPM package for Red Hat and SuSE platforms
- As an executable file with an installation script

Integrating the HASP HL Daemon in your application's installer

You can integrate the HASP HL daemon within your software's installation program.

You need to ensure that your installation program follows the steps outlined below.

1. The required HASP HL RPM packages or installation executables are copied to a temporary folder.
2. Run the required installation script or command.

For a more detailed description on installing the HASP HL daemon, refer to"Installing HASP HL under Linux" (page 41).

Chapter 12
HASP License Manager

This chapter describes the HASP License Manager. It includes the following topics:

- HASP License Manager for Windows/Mac/Linux
- Configuring the HASP License Manager
- Configuring applications protected with HASP HL Net
- Adjusting the HASP HL Net environment

Overview of HASP License Manager

The HASP License Manager maintains a login table which lists all the protected applications that have logged into a deployed HASP HL Net. The table identifies each protected application, and the station that activated the application. An application and its station are listed in the login table until the application logs out from the HASP HL Net.

The HASP License Manager uses the login table to keep track of simultaneous usage of a protected application within a networking environment. It ensures that the number of stations accessing the protected feature or application does not exceed the maximum number specified in the license for the feature as licensed by the software vendor. By default, the login table can track up to 250 applications.

HASP License Manager for Windows

The HASP License Manager for Windows is available as:

- An application for Windows 2000/XP/ Server 2003
- A service for Windows 2000/XP/Server 2003

The HASP License Manager for Windows can communicate through the TCP/IP and IPX protocols. The protocols can be loaded and unloaded using the HASP License Manager graphical user interface or settings in the *nhsrv.ini* configuration file.

The NetBIOS protocol is only available for applications protected with the HASP4 system.

Distributing the HASP License Manager for Windows

If you have protected your software to work with HASP HL Net keys, you should distribute the HASP License Manager to your end users. It is recommended that you also distribute Aladdin Monitor. For more information, refer to "Aladdin Monitor" (page 181).

There are two ways to distribute the HASP License Manager.

- You can distribute the setup file — *lmsetup.exe* — separately to your customers. See the next section for specific installation instructions. This file is available in your HASP HL installation CD under the following directory:

`Windows/Installed/Redistribute/LM/`

- You can use the HASP License Manager Installation API to include the HASP License Manager service as part of your application's installation program. This custom API is documented in separately on the HASP License Manager help file.

Installing HASP License Manager under Windows

Both types of HASP License Managers can be installed with the setup file *lmsetup.exe*.

Install the HASP License Manager on the station to which the HASP HL Net key is connected.

The installation can be customized using the following methods:

- Use the configuration file *nhsrvw32.ini*, see "HASP License Manager Configuration Settings" (page 163).
- Use the License Manager Installation API (only Win32 service), see "Settings for the IPX Protocol (Win32 only)" (page 166).

On a Windows 2000/XP/ 2003 Station

The HASP License Manager for Windows 2000/XP/2003 is *nhsrvice.exe*. Use the setup file *lmsetup.exe* to install it.

1. Install the HASP HL device driver and connect the HASP HL Net key to a station.
2. Install the HASP License Manager by running *lmsetup.exe* from your HASP HL installation CD and follow the instructions of the installation wizard. As installation type, select **Service**.

You can also integrate the HASP License Manager service installation into your application by using samples for the HASP License Manager Install API, which can be found in the following directory: *Windows/Installed/Samples/LM Install*.

Activating and Deactivating HASP License Manager

HASP License Manager Application

To activate the HASP License Manager, select the application from the **Start** menu or Windows Explorer. The HASP License Manager application is always active when any protocol is loaded and a HASP HL Net is connected.

To deactivate the application, select **Exit** from the main menu.

HASP License Manager Service

To activate and deactivate the HASP License Manager service, use the Windows Service administration in the Control Panel.

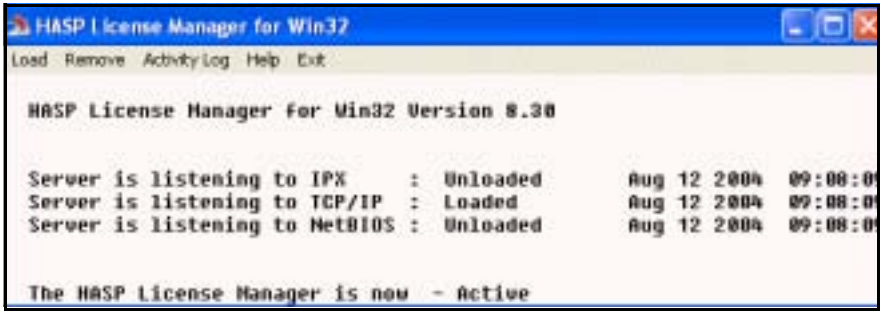
You can also use Aladdin Monitor to start and stop the HASP License Manager service.

Operating the HASP License Manager

You can operate the HASP License Manager using the graphical user interface.

To open the main window of the graphical user interface, double-click the application's icon in the system tray.

Figure 12.1 HASP License Manager GUI



The HASP License Manager main window displays the following information:

- HASP License Manager version number
- Status of each protocol (**loaded**, **unloaded**, or **failed to load**) and the date and time of the last change of status
- Status of the HASP License Manager (active or not active)

You can close the HASP License Manager main window by clicking the close button at the right corner of the title bar. The HASP License Manager continues running, and the icon remains in the system tray.

To exit the program, choose **Exit** from the menu bar.

> **!** If the HASP License Manager is installed as a Windows NT service, you cannot exit using this menu option.

Loading Protocols

To enable a protocol, select it from the **Load** menu. You can only enable protocols which have been installed on the computer.

Unloading Protocols

To disable a protocol, select it from the **Remove** menu.

Viewing the Activity Log

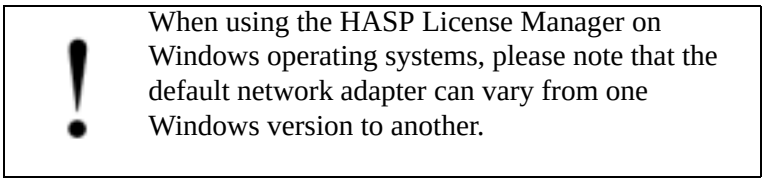
To view a log of the HASP License Manager activities, select **Activity Log** from the menu bar. The **Activity Log** window is opened.

To view the log for a specific protocol, select the protocol from the drop-down list.

Multiple Network Adapters

The HASP License Manager binds itself to the default Windows network adapter that is usually the first available network adapter.

To allow the HASP License Manager to serve requests arriving at other network adapters on a multi-homed system, **IP Forwarding** must be enabled within the Windows networking configuration.



HASP License Manager for Mac

The HASP License Manager for Mac is available for Mac OS X. It can communicate through the TCP/IP protocols.

The HASP License Manager for Mac consists of a daemon and a graphical user interface. The HASP License Manager for Mac can be operated by using the graphical user interface. You can also operate the daemon from the command line.

Distributing the HASP License Manager for Mac

The following directory on the HASP HL installation CD contains all software required for distributing HASP License Manager to your end users:

MacOS/Installed/Redistribute/LM/

There are four methods for distributing the HASP License Manager:

- An installation image which can easily be downloaded by your end users from your FTP site.
- An installation package that can be included as part of a multi-package installation.
- Source files that can be integrated to a single package installation.
- Scripts sent to end users with instruction on how to install the daemon through a command-line shell.

All four methods are included in folders within the directory specified above. Each folder contains a *readme.txt* file containing detailed information on usage and installation procedures.

Activating and Deactivating HASP License Manager

To activate the HASP License Manager, start the application from the applications menu and start the daemon by choosing **Start Daemon** in the application window. Alternatively you can start the daemon using the installation script.

To load the HASP License Manager automatically, activate the **Activate in system startup** option.

Operating HASP License Manager

You can operate the HASP License Manager using the graphical user interface. You can also operate it from the command-line tool, see "Switches for the HASP License Manager" (page 162).

The HASP License Manager for Mac displays the following information:

- Server name and IP address of the server
- Available switches
- If the daemon is activated during system start up
- The daemon status

The following options are available:

- Setting switches (only when the daemon process is not running)
- Starting and stopping the daemon
- Activating the daemon at start up

Setting a Server Name

You can assign up to six server names to the HASP License Manager.

To assign a server name:

1. Stop the daemon if it has already been started.
2. Activate the SRVNAMES option.
3. Enter up to six names — separate the names with semicolons, colons, or spaces.

The names are assigned once the daemon is started.

Avoid using non-ASCII characters for server names, since their codes differ form system to system. Server names are not case-sensitive. You cannot assign server names to a running daemon.

Setting a Configuration File

You can configure the HASP License Manager for Mac using a configuration file. To set name and path of the configuration file, activate the **CFGFILE** option and enter the path and name. For information about the configuration file, refer to "HASP License Manager Configuration Settings" (page 163).

Starting and Stopping the Daemon

To start and stop the daemon, use the buttons in the application window.

Activating the Daemon Automatically

The daemon can be automatically activated at system startup. To do this, enable the **Activate in system startup** option.

HASP License Manager for Linux

The HASP License Manager (LM) for Linux is available for all distributions of Linux. RPM packages are however, only available for the following Linux distributions:

- Red Hat 8 and Red Hat 9
- SuSE 8.x and SuSE 9.x.

The HASP HL LM for Linux consists of a daemon.

> The HASP License Manager for Linux was not available as this manual went to print. Please consult your local HASP representative for information on release dates and other updates.

Distributing the HASP License Manager for Linux

The following directory on the HASP HL installation CD contains all software required for distributing HASP License Manager to your end users:

Linux/Installed/Redistribute/LM/

There are two methods for distributing the LM.

- RPM packages, see the next section for installation instructions.
- Scripts sent to end users with instruction on how to install the daemon through a command-line shell.

All two options are included in folders within the directory specified above. Each folder contains a *readme.txt* file containing detailed information on usage.

Installing HASP License Manager

You can automatically install the HASP HL LM for Linux using RPM packages for the distributions listed above.

To install HASP HL LM on SuSE 8.x or 9.x

Use the following SuSE RPM package:

- rpm -i HASP HLlm-suse-8.30-1.i386.rpm

To install HASP HL LM on RedHat 8 or 9

Use the following Red Hat RPM package:

- rpm -i HASP HLlm-redhat-8.30-1.i386.rpm

To install HASP HL LM on other Linux Distribution

You must manually install the HASP License Manager.

1. Unpack the archive using:
   `tar-xzf [path/]linuxlm_8_30.tar.gz`
   The ‘linux-HASP HLlm_8_30’ directory is created.
2. Change to this directory and execute as root:
   `./dinst`
   This command installs the HASP License Manager and configures the system to automatically start the daemon at system boot.

Activating and Deactivating the HASP License Manager

If properly installed, the HASP License Manager should automatically be activated once the system is rebooted.

To deactivate the HASP License Manager, you must uninstall the daemon. For example, to uninstall the HASP License Manager running on Red Hat 7.3 enter the following:

```
rpm -e HASP HLlm-redhat
```

Customizing the HASP License Manager

When installing and operating the HASP License Manager you may want to adapt it to a particular network environment. You can use one the following methods:

- Operate the HASP License Manager with switches.
- Use the configuration file *nhsrv.ini*.
- Use the License Manager Installation API (Win32 only).

Switches for the HASP License Manager

The HASP License Manager can be activated with various switches. These commands instruct the HASP HL system on which protocols to use, and how to serve the HASP HL Net clients.

Table 12.1 HASP License Manager Switches

Switch	Explanation	Linux	Windows	Mac
-c	Specifies the location of the configuration file for the HASP License Manager.	yes	no	yes
-help	Displays a list of available switches.	yes	no	yes

HASP License Manager Configuration Settings

To configure the HASP License Manager, use the *nhsrv.ini* configuration file. A configurable sample of *nhsrv.ini* is included in the **Redistribute** folder on your HASP HL installation.

Search Order

You can place *nhsrv.ini* and the executable of the HASP License Manager in the same directory or in any other location according to the *nhsrv.ini* search order described in Table 12.2.

Table 12.2 Search Order for *nhsrv.ini*

Operating System	Search Order
Windows 2000/XP/ Server 2003	Executable file directory Current directory Windows 32-bit system directory Windows 16-bit system directory Windows directory Path
Linux	To use a configuration file, you must set the name and the path for the configuration file using the `-c` switch.
Mac	To use a configuration file, you must set the name and the path for the configuration file using the `-c` switch.

Server settings

Server parameters are specified in the *nhsrv.ini*.file and not in the command line. This way, parameter specification for the License Manager service is easier, and configuration is simplified and consolidated within a single file.

For Win32 platforms, the *nhsrv.ini*. should reside in the directory from which the program executable is deployed. For Linux and Mac platforms there is no default storage location for the file; you must specify which configuration file to use. For example:

./HASP HLlm -c /etc/nhsrv.ini

nhsrv.ini. Settings

Boolean switches may have the following values:

Table 12.3 Boolean Values for HASP HL LM *nhsrv.ini*

1	yes	true	enabled
0	no	false	disabled

Global LM Settings

You can fine tune settings for the HASP License Manager through its configuration file, *nhsrv.ini*. The HASP License Manager configuration file includes the [NHS_SERVER] section which is used to set global LM settings. The following is a list of keywords in the [NHS_SERVER] section:

Keyword `NHS_USERLIST`

Description Maximum number of concurrent logins to LM. Maximum number is 65520.

Default 250

Example `NHS_USERLIST =1000`

Keyword `NHS_SERVERNAMES`

Description Server name to match the name a client requests. Maximum of 6 names. Maximum 7 characters per name. Multiple names separated by commas.

Default none

Example `NHS_SERVERNAMES= cad, 3242e3`

Keyword `NHS_HIGHPRIORITY`

Description Runs LM at high priority. Default runs the LM at normal priority. Switch applies only to Win32. When this switch is set to yes, check to see how performance of other services (file server, web server, etc.) running on the same machine are affected.

Default no – runs at normal priority.

Example `NHS_HIGHPRIORITY= no`

Settings for the IP Protocol

You can fine tune settings for the HASP License Manager through the configuration file, *nhsrv.ini*. This file includes the [NHS_IP] section which is used to define settings for the IP protocol. The following is a list of keywords in the [NHS_IP] section:

Keyword `NHS_USE_UDP`

Description Setting can be either enabled or disabled.

Default enabled

Example `NHS_USE_UDP=enabled`

Keyword NHS_IP_PORTNUM

Description IP port number. This switch applies only to Win32. Port number 475 is the exclusively registered IANA number for Aladdin LM.

Default 475

Example NHS_IP_PORTNUM=475

Clients must use the same port!

Keyword NHS_IP_LIMIT

Description Specifies the range stations which are allowed to access the currently activated HASP HL LM. The last byte may be a range. Multiple entries should be separated by commas. The list may be split through several lines. The following wildcard (asterisk) byte combinations are possible: 4th,4th,3rd or 4th,3rd,2nd. An additional bitmask can be specified as the number of single bits (e.g. 10.0.0.0/8).

Default none

Example

```
NHS_IP_LIMIT = 10.242.18-99,10.1.1.9/16,
10.25.0.0/24,192.0.0*,194.0*,*,11.*,*,
*10.24.7.8-12/30,10.24.2.17
```

Settings for the IPX Protocol (Win32 only)

You can fine tune settings for the HASP License Manager through the configuration file, *nhsrv.ini*. The file includes the [NHS_IPX] section which is used to define settings for the IPX protocol. The following is a list of keywords in the [NHS_IPX] section:

Keyword NHS_USE_IPX

Description Setting can be either enabled or disabled.

Default enabled

Example `NHS_USE_IPX= yes`

Keyword

Keyword `NHS_ADDRPATH`

Description Path to HASP HLaddr.dat file. The IPX address of the current LM is written to HASP HLaddr.dat in the specified directory. Clients can specify this file in their netHASP.ini file.

Default current directory

Example `NHS_ADDRPATH=c:\temp`

Keyword `NHS_APPENDADDR`

Description Appends address data to the hapaddr.dat. If enabled, the current LM's address is added to an existing HASP HLaddr.dat. This is particularly useful when multiple HASP License Managers exist. All the LMs can be searched by the client. The LM does not search for duplicates when adding an address.

Default replace

Example `NHS_APPENDADDR=no`

Keyword NHS_USESAP

Description Setting can be enabled or disabled to allow the HASP HL LM to announce itself to the network through the Service Advertising Protocol (SAP). SAP enables clients to find the LM for different subnets. IPX is normally configured to run a virtual subnet on Win 2000/XP/Server 2003 machines, so SAP is essential in enabling clients to find the LM.

Default enabled

Example NHS_USESAP=enabled

Keyword NHS_IPX_SOCKETNUM

Description The IPX socket number. All clients must use the same default socket number. The number should not be altered. Note: clients must use the same port!

Default 0x7483

Example NHS_IPX_SOCKETNUM= 0x7483

Settings for the NetBIOS Protocol

You can fine tune settings for the HASP License Manager through the configuration file, *nhsrv.ini*. The file includes the the [NHS_NETBIOS] section which is used to define settings for the NetBIOS protocol. The following is a list of keywords in the [NHS_NETBIOS] section:

Keyword NHS_USE_NETBIOS

Description Setting can be enabled or disabled. If you are certain that you do not need NetBIOS for LM communication, disable the switch to save network and memory resources.

Default enabled

Example NHS_USE_NETBIOS=enabled

Keyword `NHS_NBNAME`

Description Use switch to alter name. Note: Clients must use the same name which must be unused in your NetBIOS name space. Names must follow NetBIOS naming conventions.

Do not use this switch unless you are certain that you need to specify a new NetBIOS name.

Default enabled

Example `NHS_NBNAME=MyNBName`

Keyword `NHS_USE_LUNA_NUMS`

Description Only included so as to be compatible with older versions.

Default all (automatic)

Example `NHS_USE_LUNA_NUMS=3,0,7,2`

Configuring HASP HL Net Clients

This section describes how an application protected for HASP HL Net — the HASP HL Net client — can be configured through a configuration file.

When the client finds the relevant configuration file, it reads and uses the information contained in the file. If not, default values are used.

In the HASP HL Net client configuration file you can fine tune how the client searches for the HASP License Manager.

The default filename of the HASP HL Net configuration file is *nethasp.ini*. A copy of *nethasp.ini* is included in the directory containing the HASP License Manager. If you change the name of the configuration file, you must implement the new name when protecting the application with HASP HL Envelope or the HASP HL API.

Search Sequence for Configuration File

The search sequence for the file depends on the operating system and the type of application.

The protected application searches for the configuration file after the first login call. For more information refer to "The HASP HL API Login Function" (page 61). The application undertakes the following search sequence.

Table 12.4 *nethasp.ini* Configuration File Search Order

Operating System	Search Sequence
Win32	Executable file directory → Current directory → Windows system directory → Windows directory → Path
Mac OS X (Carbon)	Current directory
Mac OS X	Current directory → Home directory of the current user → /etc. directory

Under Mac OS X, *nethasp.ini* is searched without a leading period. If you are using a case-sensitive system on Mac OS X, make sure that the filename *nethasp.ini* is in lowercase.

Sections in the Configuration File

The HASP HL Net client configuration file consists of three optional sections:

- `[NH_COMMON]` for general settings
- `[NH_IPX]` for the IPX protocol
- `[NH_TCPIP]` for the TCP/IP protocol

The `[NH_COMMON]` section contains global settings for all configuration file sections. The other two sections contain settings which fine tune operations for the specific protocol.

Specifying Keywords

In each section, you can specify either general or section-specific keywords. If you set a general keyword in one of the three protocol sections, you override the setting in the [NH_COMMON] section (for that protocol only).

Use the section-specific keywords to adjust additional settings for a particular protocol.

API and Envelope settings override configuration file settings.

Every line of the HASP configuration file you receive with the HASP HL software is preceded by a semicolon (;). To use a line, remove the semicolon. If you want to add comments, precede them with a semicolon.

Names of the HASP HL Net configuration files and their keywords are not case-sensitive — exception being names under Mac OS X when a case-sensitive file system is used.

The following sections describe each section in the HASP HL Net client configuration file. For each keyword, the possible values and a short description are included. Default values are listed when applicable.

[NH_COMMON]

Section specific for [NH_COMMON]

Keyword	`nh_ipx`
Description	Enables the use of the IPX protocol.
Possible values	enabled, disabled
Keyword	`nh_tcpip`
Description	Enables the use of the TCP/IP protocol.
Possible values	enabled, disabled

General keywords for [NH_COMMON]

Keyword	`nh_session`
Description	Sets the maximum length of time during which the protected application attempts to communicate with the HASP License Manager.
Possible values	numerical value
Default	2 seconds
Keyword	`nh_send_rcv`
Description	Sets the maximum length of time for the HASP License Manager to either send or receive a package.
Possible values	numerical value
Default	1 second

[NH_IPX]

Section specific keywords for [NH_IPX]

Keyword	nh_use_bindery
Description	Enables the IPX protocol to work with Novell's BINDERY.
Possible values	enabled,disabled
Default	disabled
Keyword	nh_use_broadcast
Description	Enables the use of the IPX broadcast mechanism.
Possible values	enabled, disabled
Default	enabled
Keyword	nh_bc_socket_num
Description	Sets the socket number for the broadcast mechanism. The number is hexadecimal.
Possible values	hexadecimal value
Default	7483H
Keyword	nh_use_int
Description	2F_NEW means that the IPX protocol only uses interrupt 2Fh. 7F_OLD means that the IPX protocol only uses the 7Ah interrupt.
Possible values	2F_NEW, 7F_OLD
Default	2F_NEW
Keyword	nh_server_name

Description	Communicates with the HASP License Manager under a specified name.
Possible values	`<name1>,<name2>,...`
Keyword	`nh_search_method`
Description	Determines if the protected application communicates with HASP License Manager on a local network, or with any HASP License Manager on the Internet.
Possible values	`localnet, internet`
Default	`internet`
Keyword	`nh_datfile_path`
Description	Specifies the location of the HASP License Manager address file.

General keywords for `[NH_IPX]`

Keyword	`nh_session`
Description	Sets the maximum length of time during which the protected application tries to communicate with the HASP License Manager.
Possible values	`numerical value`
Default	`2 seconds`
Keyword	`nh_send_rcv`
Description	Sets the maximum length of time for the HASP License Manager to either send or receive a package.
Possible values	numerical value
Default	1 second

[NH_TCPIP]

Section-specific keywords for [NH_TCPIP]

Keyword	nh_server_addr
Description	Sets IP addresses for all the searchable HASP License Managers. Unlimited number of addresses and multiple lines are possible.
Possible values	<address1>,<address2>
Examples	IP address: 192.114.176.65 Local hostname: ftp.aladdin.com
Keyword	nh_server_name
Description	Communicates with the HASP License Manager under a specified name. Each name can contain up to 7 characters and you can specify up to 6 names.
Possible values	<name1>,<name2>
Keyword	nh_use_broadcast
Description	Enables the use of the UDP broadcast mechanism.
Possible values	enabled, disabled
Default	enabled

General Keywords for [NH_TCPIP]

Keyword	nh_session
Description	Sets the maximum length of time during which the protected application tries to communicate with the HASP License Manager.
Possible values	numerical value
Default	2 seconds

Keyword	`nh_send_rcv`
Description	Sets the maximum length of time for the HASP License Manager to either send or receive a package.
Possible values	numerical value
Default	1 second

Adjusting the HASP HL Net Environment

This section describes additional HASP License Manager settings and *nethasp.ini* keywords to accommodate the deployment of a HASP HL Net within a specific network environment.

Defining the Range of Stations under IPX

With IPX, you can allow specific stations on a different segment to access the HASP License Manager.

To allow access from a different segment:

1. Load the HASP License Manager with the **-ipxnosap** switch.

This ensures that the address of the HASP License Manager is not advertised using the SAP mechanism, and is advertised in the HASP HL Net *newhaddr.dat* address file.

2. Edit the *nethasp.ini* file as follows:

- In the `[NH_COMMON]` section, set `NH_IPX = Enabled`
- In the `[NH_IPX]` section, set `NH_USE_BROADCAST = Disabled`
- In the `[NH_IPX]` section, set `NH_USE_BINDERY = Disabled`

These settings instruct the protected application running on stations in the range to search for the address file and read the address of the HASP License Manager.

3. Copy the protected application and the *nethasp.ini* file to the same directory.

Defining the Range of Stations under TCP/IP

There are two ways of defining the range of stations under TCP/IP. You can either specify the range of stations that the HASP License Manager serves, or you can specify that the range of stations search for a particular HASP License Manager.

Specifying the Range Using *nhsrv.ini*

The HASP License Managers for Windows, Win32 and Mac can read a *nhsrv.ini* configuration file. You can edit this file to specify the range of stations the HASP License Manager serves under TCP/IP.

To specify the range of stations

Edit the *nhsrv.ini* file as follows:

- In the `[NHS_SERVER]` section, `NHS_IP_LIMIT = <ipaddr> [,<ipaddr....]`

Sample Formats for <ipaddr>

When you specify the range of stations using *nhsrv.ini*, you can use any of the following formats:

- 10.1.2.3

The HASP License Manager serves only the station with the specified IP address.

- 10.1.2.*

The HASP License Manager serves only stations that match the specified IP address mask, i.e. 10.1.2.0 through 10.1.2.255.

- 10.1.*.*

The HASP License Manager serves only stations that match the specified IP address mask, i.e. 10.1.0.0 through 10.1.255.255.

Limiting station access to the HASP HL Net in a TCP/IP-based network:

1. Edit *nhsrv.ini* and set the range of stations.
2. Copy *nhsrv.ini* to a location accessible by the HASP License Manager.

Specifying the range using *nethasp.ini*

You can edit the HASP HL Net configuration file to specify that the HASP License Manager (according to address) should search a range of stations.

Specifying a range of stations

1. Edit the *nethasp.ini* file: In the `[NH_TCPIP]` section, set `NH_SERVER_ADDRESS= <address of` HASP License Manager`>`
2. Copy the *nethasp.ini* to a location accessible only to the desired range of stations.

Adapting the Time-out Length

The HASP License Manager cannot serve more than one request at a time. The time-out length determines how long a protected application repeatedly tries to access the HASP License Manager before giving up.

In almost all networks, the default time-out values are sufficient, so you only need to change the default values in networks that have a HASP HL Net connected to a slow or busy station.

Defining time-out length

In the appropriate section of the *nethasp.ini* file, set:

```
NH_SESSION = <m>
NH_SEND_RCV = <n>
```


where m and n are measured in seconds. By default, m is 2 seconds and n is 1 second.

Defining the Number of Protected Applications Served

With the HASP License Manager, you can change the default number of stations served. By default the HASP License Manager can serve a maximum of 250 (Win32, Mac and Linux) stations.

The HASP License Manager allocates memory space for the maximum number of protected applications. If necessary, you can save memory space by changing this default value.

Changing the default memory space allocation

Load the HASP License Manager with the switch:

nhsrvw32 -userlist = n

where n is the number of stations it serves.

The **-userlist** switch is only available for Win32.

Chapter 13
Aladdin Monitor

This chapter describes the Aladdin Monitor utility. It includes the following topics:

- Installing Aladdin Monitor
- Settings for the Aladdin Monitor utility
- Monitoring the HASP License Manager
- Checking HASP HL Net keys

Aladdin Monitor permits centralized administration of the HASP License Manager applications and the HASP HL Net keys.

Aladdin Monitor is available for the following environments: Windows 98SE/ME/NT/2000/XP/Server 2003. It communicates via TCP/IP and IPX protocols.

Supply your customers with Aladdin Monitor together with:

- Protected software
- HASP HL Net keys
- HASP License Manager

The Aladdin Monitor is distributed with an online Help system.

You can modify the behavior of Aladdin Monitor with a HASP HL Net client configuration file — (*nethasp.ini*). For more details, refer to "Configuring HASP HL Net Clients" (page 169).

The following functionality is provided by the utility:

- Checking the properties of the HASP License Manager
- Checking HASP HL Net keys
- Starting and stopping the HASP License Manager service

Installing Aladdin Monitor

You can install Aladdin Monitor on any station in the network. It is not necessary to install a HASP License Manager on the same station.

To install Aladdin Monitor, use the *aksmon32.exe* installation utility. Once launched, follow the on-screen instructions.

Settings for Aladdin Monitor

You can adapt the following Aladdin Monitor settings to meet your requirements:

- The display language — English or German
- The refresh frequency for the dialog box — default setting is every 2 seconds
- The frequency of network queries — default setting is every 3 minutes
- You can determine whether Aladdin Monitor runs in HASP mode, Hardlock mode or both
- Determine whether or not you want to use the *nethasp.ini* configuration file

To change the settings, select **Settings** from the **File** menu. The changes become active after the program is restarted.

Monitoring the HASP License Manager

In the left panel of the window, under **Aladdin Network Resources,** click the HASP License Manager for which you want to check the login information.

If no HASP License Manager entries are displayed, first double-click the **HASP License Manager** folder or refresh the view by selecting **File/Rescan**.

The HASP License Manager information is displayed in the right panel of the window.

HASP License Managers only listening to NetBIOS are not recognized by Aladdin Monitor.

The following information about the selected HASP License Manager is displayed:

- General information about the selected HASP License Manager (Table 13.1)
- Information about the HASP HL Net keys being managed (Table 13.2)

Table 13.1 HASP License Manager Information

Field	Explanation
Name	Name of the computer on which the HASP License Manager is running
Version	Version of the HASP License Manager
IP	IP address of the computer
IPX	IPX address of the computer
LM Type	The platform for which HASP License Manager is configured
TCP/IP, IPX	The protocols currently being used

Table 13.2 HASP Key Information

Field	Explanation
HASP #	ID of the HASP network key
Key Type	HASP key type: HASP and older, HASP4, or HASP HL
Key Model	Maximum possible number of licenses
Current Stations	Stations currently logged in

Checking HASP HL Keys

Select the HASP HL key that appears below the HASP License Manager folder under Aladdin Network resources in the left panel of the window. The HASP HL key can only be checked, if a login has been performed.

If the key is not displayed, double-click the HASP License Manager that makes the key available, or refresh the view by selecting **File/Rescan**.

The HASP HL information is displayed in the right panel of the window.

The following information about the selected HASP HL key is displayed:

- General information about the HASP HL key (Table 13.3)
- An overview of the programs (Table 13.4)
- An overview of logins for the individual programs (Table 13.5)

Table 13.3 HASP HL Information

Field	Explanation
HASP #	Cumulative number of the HASP key
Key type	Type of HASP Key: HASP HL, HASP or HASP4

Table 13.4 Program Table

Field	Explanation
Program No.	Number representing the protected program
Current Stations	Stations currently logged in
Maximum Stations	Maximum possible number of stations
Activations	Maximum number of program activations

Table 13.5 Login Table

Field	Explanation
No.	Cumulative number of the login
Login ID	Address under which the station logged in
Host Name	Name of machine to which the key is connected
Protocol	Protocol used
Timeout	Time which must elapse without activity until the login entry is deleted or cancelled (in seconds)

The HASP License Manager Service

The HASP License Manager service enables you to administer HASP HL Net keys on an NT workstation.

You can use the Aladdin Monitor to start and stop the HASP License Manager service on the local computer.

Starting the Service

Select **Start HASP LM Service** in the **HASP LM Service** menu or the **Services/HASP** menu. Alternatively, you can use the traffic light symbol. The service is started and can now make locally connected HASP HL Net keys available within the network.

Alternatively, you can start the service using the context-sensitive menu. To do this, right-click the **HASP License Manager** folder and select **Start HASP LM**.

Stopping the Service

Select **Stop HASP LM Service** in the **HASP LM Service** menu or the **Services/HASP** menu. Alternatively, you can use the traffic light symbol.

The service is stopped. The view is then refreshed. This may take some time since it involves scanning the entire network.

Alternatively, you can stop the service using the context-sensitive menu. To do this, right-click the **HASP License Manager** folder and select **Stop HASP LM**.

Chapter 14
Diagnosing HASP HL Keys

This chapter describes how Aladdin DiagnostiX and Aladdin DiagnostiX Memory Beamer are used to collect information on deployed HASP HL keys. It includes the following topics:

- Delivering Aladdin DiagnostiX to your customers
- Diagnosing HASP HL keys
- Creating reports on deployed HASP HL keys
- Linking Aladdin DiagnostiX to external reporting tools
- Using Aladdin DiagnostiX Memory Beamer

The Aladdin DiagnostiX utility collects information on deployed HASP HL keys and the systems on which they are running. This feedback mechanism helps provide solutions to customers encountering problems related to the protected application.

Aladdin DiagnostiX Functionality

Your customers can use the Aladdin DiagnostiX utility to do any of the following:

- Check for a HASP HL key
- Create a report file that contains data on Aladdin devices and other relevant system information
- Download an updated HASP HL driver if required

Aladdin DiagnostiX is available for the following environments: Windows 98SE/ME and Windows NT/2000/XP/Server 2003.

When your customers run the Aladdin DiagnostiX utility, instruct them to select the **Check HASP HL** tool to check for a HASP HL key. In addition to HASP HL data, your customers can use Aladdin DiagnostiX to generate reports containing vital information about their systems. These reports are delivered to technical support for further analysis.

The utility diagnoses your system at startup with the **System Info** diagnostic tool. Results are displayed in the workspace as part of the **System Info** screen.

Distributing Aladdin DiagnostiX

You should distribute Aladdin DiagnostiX to your customers. The installation of Aladdin DiagnostiX at the customer's site is a simple process. Your customers simply click *aksdiag32.exe* and follow the on-screen instructions.

A copy of *aksdiag32.exe* is provided in the **Redistribute** folder of your HASP HL installation and the HASP HLinstallation CD. Deliver this file by e-mail or burn it on a CD and ship to your customers.

The HASP HL vendor codes are required to access and communicate with the keys. To maintain confidentiality we recommend sending your customers a customized DLL containing the vendor code data in a secured manner. To create a customized DLL, refer to "Sending Customized DLLs to Your Customers" (page 198).

Diagnosing HASP HL Keys

The Aladdin DiagnostiX utility includes the **Check HASP HL** tool to access and retrieve information on deployed HASP HL keys.

Checking for a HASP HL Key

To check for a HASP HL key, select the **HASP HL** icon in the **Diagnostic Tools** pane. The **Check HASP HL** screen is displayed in the workspace.

To check a HASP HL key

Follow the procedure below to check for a HASP HL key.

1. Select a **Search Mode** from the **Check HASP HL** screen. The default setting is **Local and Remote**.
2. Specify the Vendor Code for the HASP HL key.
3. Specify the Program number. We recommend selecting Default (0) as this feature is always available if the HASP HL key is connected.
4. If you want to read and save the memory content of the HASP HL key, check **Save update information when checking key.**
5. Click **Check Key**.
6. Details of the key access are displayed in the **Key Access History** panel in Aladdin DiagnostiX.

Key Access History Panel

The **Key Access History** panel records and tabulates all attempts to access HASP HL keys. Key access information is displayed sequentially, showing the most recent access first. Table 14.1 details the **Key Access History** panel.

Table 14.1 Aladdin DiagnostiX Key Access History panel

Column	Value	Description
Access Mode	local	The key was found on a local port.
	remote	The key was found on a remote port.
Prog No.	Program number used to login to HASP HL key.	
Port/IP	Displays the number of the port to which the HASP HL key is connected. If a HASP HL was found and logged in on a remote port using the HASP License Manager, the IP address is displayed. If the HASP HL key cannot be accessed, **(n/a)** is displayed.	
Key Type	The type of HASP HL key detected.	
HASP ID	ID number of the detected key.	
Addit. Info	Additional information related to error number displayed above.	

Configuring HASP HL *nethasp.ini* files

Aladdin DiagnostiX enables your customers to quickly configure and create HASP HL *nethasp.ini* files and files, thus speeding up communications between their protected applications and HASP HL Net keys.

To access the **nethasp.ini Configuration** screen:

1. Double-click the **HASP** icon in the **Diagnostic Tools** pane. The HASP **nethasp.ini** icon appears below the HASP icon.
2. Click on the HASP **nethasp.ini** icon.
   The **nethasp.ini Configuration** screen is displayed in the workspace.

Your customers have three options for creating a *nethasp.ini* file.

- Using an **automated** file. The file will is created based on information collected at startup. Information contained in the file is then used to access servers. The first suitable protocol is used in the following order: IP and then IPX. Only IP server names found at startup are used.
- Using a **default** file. In this option, your customers can specify any of the following to be included in the *.ini* file.
  - IP
  - IPX

If this option is selected, no pre-defined settings for server names are included in the default *.ini* file.

- Using a **customized** file. Aladdin DiagnostiX enables your customers to set particular local and server protocols.

Configuring *nethasp.ini* files

Use the following procedure to configure an *.ini* file using Aladdin DiagnostiX:

1. Select one of the three methods provided to create the file. If you select the **automated** options, skip steps 2 - 3 and proceed to step 4.
2. If you selected the **default** option in step 1, check any, or all, of the protocols provided. Skip to step 4.

3. If you selected the **customized** option in step 1, use the fields provided to define IP and IPX servers. You can manually use command-line parameters or click the corresponding browse button to view and select from a list of available options.
4. Enter the name of the output directory for the *.ini* file in the **Set output directory** field. You can also use the browse button to navigate to the desired directory.
5. Click **Create**.

Creating Reports on HASP HL keys

Your customers can use Aladdin DiagnostiX to create reports containing information on the following:

- HASP HL devices
- System information
- Information pooled by external reporting tools
- Memory read from HASP HL keys

If your customers experience a problem with their HASP HL device or have difficulties in accessing the protected application, they can e-mail the reports to you or to the local Aladdin technical support staff.

Create Report Settings

The content and format of the reports generated by Aladdin DiagnostiX is defined in the Aladdin DiagnostiX Settings screen. Use the **Create Report** tab to define your settings.

The settings that appear in the **Create Report** tab include:

Report format: offers three format possibilities — XML, HTML, and TEXT.

If you select either the HTML or Text options, an additional XML file will be created containing the read memory of any key detected.

The Aladdin DiagnostiX reporting feature may generate multiple files. To zip the generated files, check **Zip all output files**.

To include information on Win16 and DOS sub-systems in the generated reports, check the **Include Win16/DOS** box. Aladdin DiagnostiX generates two separate report files for each sub-system.

To create a report:

1. Review the settings outlined in the preceding section.
2. Use one of the following methods to create a report.
   - Click the **Create Reports** icon.
   - Select **Create Report** from the **Edit** menu.
   - [Ctrl] + **R**
   - Click the **Create Report** button on the System Info screen.

A message box appears indicating the contents of the report file and its location.

Linking to External Reporting Tools

Aladdin DiagnostiX allows you to define settings for two separate external reporting tools such as msinfo32. The reports generated by these external tools can be zipped and stored in a single file. Use the procedure below to link Aladdin DiagnostiX to external reporting tools:

1. Open the **Reporting Tools** tab in the Aladdin DiagnostiX Settings dialog.
2. Check the **Reporting tool 1** box.
   The Execute and Output File fields should be activated.
3. In the **Execute** field, either specify the command-line path to the target reporting tool, or browse for the tool using the button. Add additional command-line parameters to run the tool in ‘silent mode’ and specify the output Aladdin DiagnostiX file.
4. Use the **Output File** field to specify the name of the file generated by the external tool. Specify the path to the destination directory, or browse for the file.
5. To define settings for a second external reporting tool, check the **Reporting Tool 2** box and repeat steps 2-4.
   You have the option of setting a time limit for the external tool to generate a report. Check the appropriate box and specify a time limit. If the reporting tool does not generate its report within the allocated time frame, it will be terminated.
6. Click **OK** to exit.

Updating HASP HL Drivers

Your customers can use Aladdin DiagnostiX to update drivers for locally installed keys. These updated drivers are available from the Aladdin FTP site.

To update the drivers:

a. Click the **Driver Update** button provided in the System Info screen.

b. Follow the on-screen instructions to download and install the new driver(s).

Your updated drivers are displayed immediately in the **System Info** screen.

Aladdin DiagnostiX Memory Beamer

The Aladdin DiagnostiX Memory Beamer is a vendor utility that serves as a channel for transferring secured data between vendors and their customers. In conjunction with the Aladdin DiagnostiX tool, the Aladdin DiagnostiX Memory Beamer:

- Enables you to send customized DLLs to your customers.
- Accesses reports generated by the Aladdin DiagnostiX utility.
- Reads the update information that includes the contents of the memory in the deployed HASP HL key. The information can be saved as a C2V file to be processed by HASP HL Factory.

Aladdin Diagnostix Memory Beamer is designed for the exclusive use of software vendors and not end users.

Sending Customized DLLs to Your Customers

Use the Beamer to ‘inject’ the required vendor codes for deployed keys into customized DLL files that can later be supplied to your customers. Once available on your customers’ system, these customized DLLs provide an additional means to access a HASP HL key via the Aladdin DiagnostiX utility.

Follow the instructions below to prepare a DLL

1. Enter the path or browse for the Vendor Code file.
2. Click **Inject**.
   A message box appears confirming that the code has been ‘injected’ into the DLL.
   A DLL file should be displayed in your current directory as *custom.dll* — do not rename this file!

3. Send the DLL file to your customer with instructions to store the file in the same folder where their Aladdin DiagnostiX utility is located. Alternatively your customer can place the DLL in the folder containing the Aladdin DiagnostiX installer — the installer will automatically copy the DLL to the installation directory.

Reading Report Files

The Beamer enables you to quickly access and sort reports generated by the Aladdin DiagnostiX utility. The Beamer displays the following information based on information stored in report files sent by your customers:

- Type of HASP HL key
- Access — how was the key accessed

Reading a report file

1. Enter the path or browse for the report file. Click **Open**.
2. If the report file contains relevant HASP HL entries, the **Key Type and Access** fields should contain entries.

If at any time you wish to skip a particular report file and move on the next file, click **Skip**.

3. If the report file contains memory content of a deployed HASP HL key, the information can be saved as a C2V file to be processed by HASP HL Factory. Provide a file name for the file or use the browse button to navigate to a desired location. The default file name is *[haspid].c2v* - where *haspid* is the serial number of the deployed HASP HL key.

Appendix A
Troubleshooting

The first part of this appendix offers a checklist that can help you solve some of the most common problems you might encounter when using the HASP HL system. The second part lists specific problems you or your customers may experience, along with the solutions.

The HASP HL product line conforms to the highest standards of quality assurance. However, like any other PC peripheral device, it might not operate on certain PC configurations because of faulty equipment or improper installation. This appendix can help you in such a situation.

In addition to the information in this appendix, customers can access the Aladdin Knowledge Base at:

`http://www.hasp.com/kb2`

The Knowledge Base contains a comprehensive listing of solutions to general and specific problems. To avoid difficulties, make sure you are using current HASP HL software versions. Contact your local HASP HL representative for the latest updates or refer to Aladdin's international downloads page at:

`http://www.hasp.com/download`

If problems persist, check whether the HASP HL samples and Aladdin DiagnostiX work properly, and send the results to your local HASP HL representative.

Checklist

If a customer reports a problem, check the following list:

- When applicable, note the returned error code or message. For more information, refer to "API Status Codes" (page 242).
- Is the HASP HL connected properly to the USB port?
- Is your customer’s computer or system experiencing technical difficulties such as device manager collisions, system events, bootlog failures, etc.?
- Can Aladdin DiagnostiX access the HASP HL? Try to create a diagnostic report. Refer to "Creating Reports on HASP HL keys" (page 194) for more details.
- Does the problem occur when the protected application runs on another PC of the same model?

Problems and Solutions

Problem	HASP HL drivers do not install.
Solution	Are there older HASP device drivers installed on the machine? You should uninstall the older driver installations using the installer corresponding to the older driver version. Refer to the HASP driver documentation for details. Once the older drivers are removed, install the HASP HL drivers. For more information, refer to "Installing the HASP HL Device Drivers" (page 35).
Problem	You try using *haspdinst.exe* to install the HASP HL device driver under Windows 2000/XP/2003 but receive an error message.
Solution	Review the *haspdinst.exe* installation instructions. Refer to *"The haspdinst.exe Utility" (page 35)*. You could also try to install the drivers using the *HASPUserSetup.exe*, refer to "The HASPUserSetup.exe application" (page 35).

Problem	The protected application cannot find the HASP HL key.
Solution	Does the HASP HL LED light up? If not, this could be for one of the following reasons: **1.** The key is not connected properly to the USB port. Disconnect and reconnect after a few seconds. If the LED lights up, the application should now be able to access the key. **2.** The required device drivers are not installed. If you are running the HASP HL on a Windows platform, check for an entry for the key in the Device Manager utility. If no entry appears, you must install the drivers using one of the methods in "Installing the HASP HL Device Drivers" (page 35). **3.** Check to see if the USB port is functioning properly. Disconnect all other USB devices from their respective ports. Connect the key to another USB port. Try to use another USB device in the port where the key was not accessible.

Problem	Your application takes a long time to find the HASP HL Net on a large TCP/IP network.
Solution	It is recommended that you customize the search mechanism. Use the HASP HL Net configuration file to specify the UDP or TCP search method and to set the IP address of the HASP License Manager. By doing so, the HASP HL Net client searches for the HASP License Manager with the specified IP address, which is much faster. For more information, refer to "Configuring HASP HL Net Clients" (page 169).

Problem	You receive an error message indicating that the network is busy when the HASP HL Net communicates with the HASP License Manager.
Solution	Increase the time frame in which the HASP HL Net client waits for the answer by increasing the time-out length in the HASP HL Net. For more information, refer to "Configuring HASP HL Net Clients" (page 169).

Problem	You receive an error message indicating that the HASP License Manager was not found.
Solution	You might receive this error under TCPIP/IPX when you use the broadcast search mechanism. Increase the time-out length in the *nethasp.ini* file to 8 seconds. For more information, refer to "Adjusting the HASP HL Net Environment" (page 176). If the error message is displayed again, it could be because of one of the following reasons: **a.** A HASP License Manager was not loaded. **b.** If the TCP/IP protocol was used, the HASP License Manager is in a different subnetwork. **c.** If the IPX protocol was used, SAP is not supported. **d.** If you repeatedly receive the error message, try using another search mechanism.

Appendix B
HASP HL Glossary

Activation Counter	Licensing element indicating the number of times a feature, licensed by HASP HL, can be run.
AES	The Advanced Encryption Standard (AES) algorithm is the basis for HASP HL encryption and decryption.
Aladdin DiagnostiX	A tool used to check all the information related to HASP HL on the end user's system. The information is useful when providing technical support.
Anti-Debugging	Measures undertaken by the HASP HL system to block potential attacks that seek to undermine the protection scheme.
API Call History	A log of all calls to the HASP HL API executed using HASP HL ToolBox.
API samples	Sample applications that utilize the HASP HL API. A learning tool used for implementing the HASP HL API.
Background checks	Random checks executed by protected applications for the required HASP HL.

Backward compatibility	Ability to share data or commands with applications protected with HASP4. HASP HL backward compatibility includes the ability to read and write data, set real-time clocks, and other 'legacy' commands.
Batch	Batches are used to define features and packages, and to execute orders. Typically, a single batch is required to license all your products. Keys belonging to a particular batch have a unique encryption and decryption behavior.
Batch Code	A 5-8 character unique code that is assigned to a vendor and is printed on the HASP HL label. The code is used for ordering HASP HL keys.
Broadcast	A communication method across a network where messages are broadcast (sent) to every workstation.
Broadcast search mechanism	The search mechanism that the HASP HL client uses to find the HASP License Manager.
C2V file	Customer to vendor file. A file sent by the customer to the vendor containing information on the status of a deployed HASP HL.
Concurrent	Used in relation to HASP HL keys or licence and indicates simultaneous running of keys or licenses.
Configuration file	A file that contains configuration information for a particular program. When the program is executed, it consults the configuration file to see what parameters are in effect.
Crack(ing)	The deliberate breaking of a protection scheme.
Cross platform	An indication that functionality is available on multiple operating systems.

Daemon	A process that runs in the background on Mac and Linux platforms. Typical daemon processes perform administrative tasks for the operating system.
Decryption	A process of decoding data that has been encrypted into a secret format.
Default feature	A feature that is always available within the HASP HL key. It requires no configuration and its licensing behavior is conditioned by the connected HASP HL model.
DEMOMA	A Batch Code used for evaluation purposes with the any HASP HL program. Its corresponding Vendor Code is available in the *Vendorcodes* folder of your HASP HL installation.
Demo Vendor Code	See DEMOMA.
Encryption	The translation of data into a confidential code. Encryption is the most effective way of achieving data security. To read an encrypted file, you must have the correct encryption engine to decrypt the file.
Encryption engine	The encryption engine in the HASP HL key — based on the AES algorithm.
Encryption key	The key used for encrypting a data file to be used with HASP HL Envelope.
Encryption level	Controls the compactness of the HASP HL calls in HASP HL Envelope protection.
Envelope	See HASP HL Envelope.
Envelope template	Contains defined default protection settings and other project related data.
Expiration date	The date after which a protected application stops running.

Feature	A notable functionality of a software application that can be independently controlled by a license. A feature can be an entire application, an executable file or a specific functionality such as Print, Save or Draw.
Feature ID	A unique identifier for a HASP HL protected feature. The feature ID is assigned using the HASP HL Factory tool.
Feature Update	See update.
File filter	Defines the files that are exposed to 'on the fly' file encryption.
Functionality	The actions an application can perform. A product's functionality is used by marketers to identify product features.
Handle	A unique identifier to access the context of a HASP HL login session.
HASP	Software protection and licensing system.
HASP HL	Hardware device for the HASP HL copy protection and licensing system.
HASP HL API	An interface for inserting calls to a HASP HL device.
HASP HL Basic	A standard HASP HL key that is used to protect software. It does not have any memory functionality and therefore should not be used for licensing.
HASP HL Demo	A sample key provided for evaluating HASP HL protection and licensing software.
HASP HL Developer Kit	A kit containing software and documentation for the HASP HL system.

HASP HL Device Driver	A system component, that enables communication between the protected application and the HASP HL key.
HASP HL Envelope	A tool that wraps an application in a protective shield, ensuring that the protected application cannot run unless the required HASP HL key is connected to the computer.
HASP HL Factory	A tool used to define licensing elements and to produce licenses for HASP HL keys, thereby initializing their memory.
HASP ID Number	Unique Serial number for a HASP HL key.
HASP HL Max	A HASP HL key with 4 KB memory and the ability to store up to 112 licenses.
HASP HL Net	A HASP HL network key with 4 KB memory and the ability to store up to 112 licenses.
HASP HL Pro	A HASP HL key with 112 bytes of memory and the ability to store up to 16 licenses.
HASP HL RUS	A tool for end users to remotely update HASP HL licenses or modify the contents of its memory. See C2V file and V2C file.
HASP HL Time	A HASP HL key with 4 KB memory and a real-time clock. It can store up to 8 licenses to handle expiration dates and 104 for activation and perpetual usage.
HASP HL ToolBox	A HASP HL tool designed to familiarize users with the HASP HL API and to generate source code.
HASP License Manager	Acts as a server and monitors concurrent usage according to the licenses stored in HASP HL Net.

Idle Time	The time interval after which a station is considered inactive when accessing a HASP HL Net key.
Key	See HASP HL.
License	A digital permit stored inside a HASP HL key. It is a description of rights that can be applied to a feature.
License Manager	See HASP License Manager.
Login	Establish a session with a HASP HL.
Master HASP HL	A special HASP HL key that contains the unique codes and identifiers assigned to you by Aladdin that are used with the HASP HL system for protecting and licensing software.
Package	A licensing term referring to a set of features governed by specific licensing terms and programmed HASP HL memory. Packages are defined in HASP HL Factory.
Program Number	A unique identifier for a feature in the HASP HL system.
Real-Time Clock (RTC)	Clock available in the HASP HL Time key.
Remote Update System	A system used for secure, remote updating of deployed HASP HL keys.
Reverse Engineering	Software attacks that aim to unravel the algorithms and execution flow of the target application by tracing the compiled protected application to its source code. HASP HL implements contingency measures to ward off such attacks and prevent hackers from uncovering algorithms used inside protected software.

Status code	Error or status message returned by the HASP HL system.
Update	A file containing more current information applied to a HASP HL key which already has licensing data.
USB	Universal Serial Bus is a type of plug-in connection that is used to connect the HASP HL to desktop or laptop computers.
USB Port	Socket to which the HASP HL is connected.
UTC	Coordinated Universal Time — (UTC) — the standard time common to every place in the world. It is expressed using a 24-hour clock and uses the Gregorian calendar.
V2C file	A file sent from the vendor to the end user. This file contains data to update the license in a deployed HASP HL.
Vendor	A vendor is a company that sells software in which it has invested intellectual and capital resources, and would like to protect its investment from unauthorized and illegal usage.
Vendor Code	Contains a unique vendor identifier and all the information for the HASP HL system to find the vendor's key and communicate with it. The Vendor Code is distributed to vendors in the Master HASP HL.
Vendor private key	A 1536-bit encryption key stored in the Master HASP HL key and used to digitally sign RUS license extensions or memory updates sent to end users.
Vendor public key	A 1536-bit key used to authenticate the license or memory updates sent to end users.

Appendix C
HASP HL API Reference

This appendix is divided into three sections:

- Overview of the functions that constitute the HASP HL API
- Structural declarations and detailed information on individual HASP HL API functions
- A summary and description of all the available API return codes

HASP HL ToolBox

To properly understand the functionality of each API call, use HASP HL ToolBox. This tool enables you to test function calls, the required parameters, and to anticipate return values. HASP HL ToolBox is included in the Vendor Center suite of programs.

API Samples

Every HASP HL installation includes API samples for various programming languages. Use these samples to integrate HASP HL protection into your own code.

Every sample folder includes a HASP HL header file. Please refer to the Aladdin web site and the HASP HL installation CD for information on available samples programs for specific programming languages.

API Function Overview

Table C.1 lists the available HASP HL API functions.

Table C.1 List of HASP HL API functions

Function	Short Description
hasp_datetime_to_hasptime()	Converts a date/time value
hasp_decrypt()	Decrypts a buffer using the AES encryption algorithm
hasp_encrypt()	Encrypts a buffer using the AES encryption algorithm
hasp_free()	Frees up allocated memory resources
hasp_get_rtc()	Reads the current time from a HASP HL Time key
hasp_get_sessioninfo()	Retrieves information regarding a session context
hasp_get_size()	Retrieves the byte size of a memory file from a HASP HL key
hasp_hasptime_to_datetime()	Converts a time value
hasp_legacy_decrypt()	Decrypts a buffer using HASP4 backward compatible encryption
hasp_legacy_encrypt()	Encrypts a buffer using HASP4 backward compatible encryption
hasp_legacy_set_idletime()	Sets the HASP License Manager (LM) idle time value
hasp_legacy_set_rtc()	Backward compatability function setting time in a HASP4 Time key
hasp_login()	Logs in to a feature, establishing a session context
hasp_logout()	Logs out from a context or session
hasp_read()	Reads the memory of a HASP HL key
hasp_update()	Writes an update for a HASP HL license
hasp_write()	Writes to the memory of a HASP HL key

HASP HL API Abstract

The HASP HL API functions in this reference are listed alphabetically and apply to the C programming language interface. The information for each function includes the following:

- **Name**: The function name as it exists in the C language interface.
- **Description**: A brief description of the main purpose of the function's main purpose.
- **Syntax**: Actual function declaration in the C programming language.
- **Parameters**: Lists the parameters for the function.
- **Returns**: Lists all the possible returns associated with the execution of the function.
- **Usage notes**: More detailed information on how to use the function.

hasp_datetime_to_hasptime()

Description Converts a date /time value to hasptime —the number of elapsed second since 01/01/1970.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_datetime_to_hasptime(
          unsigned int  day,
          unsigned int  month,
          unsigned int  year,
          unsigned int  hour,
          unsigned int  minute,
          unsigned int  second,
          hasp_time_t *  time
          )
```

Parameters

day	input for day value (range 1-31)
month	input for month value (range 1-12)
year	input for year value (range 1970+)
hour	input for hour value (range 0-23)
minute	input for minute value (range 0-59)
second	input for second value (range 0-59)
time	pointer for placing time value

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_TIME	time beyond the supported value range

Usage notes The converted value is the number of seconds that have elapsed since Jan. 1, 1970. This conversion function is used in conjunction with the API functions that set or retrieve values for the real-time clock (RTC) of a HASP HL key.

See also hasp_hasptime_to_datetime()
hasp_get_rtc()
hasp_legacy_set_rtc()

hasp_decrypt()

Description Decrypts a buffer using the AES encryption algorithm.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_decrypt(
          hasp_handle_t  handle,
          void *  buffer,
          hasp_size_t  length
          )
```

Parameters

handle	handle for the session
buffer	pointer to the buffer to be decrypted
length	size (in bytes) of the buffer to be decrypted — 16 bytes minimum required

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_NOT_IMPL	feature type functionality not implemented
HASP_TOO_SHORT	decryption data length is too short
HASP_ENC_NOT_SUPP	hardware does not support decryption type
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes The function reverses the operation of the hasp_encrypt() function applied on a data buffer, returning the latter to its pre-encryption state. If the decoding operation fails, the data targeted for decryption is not affected.

See also hasp_encrypt()

hasp_encrypt()

Description Encrypts a buffer using the AES encryption algorithm.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_encrypt(
          hasp_handle_t  handle,
          void *  buffer,
          hasp_size_t  length
          )
```

Parameters

handle	handle for the session
buffer	pointer to the buffer to be encrypted
length	size (in bytes) of the buffer to be encrypted — 16 bytes minimum required

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_NOT_IMPL	feature type functionality not implemented
HASP_TOO_SHORT	encryption data length is too short
HASP_ENC_NOT_SUPP	hardware does not support encryption type
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes Encodes data using the encryption engine in the HASP HL key. The specified session handle determines which HASP HL key performs the encryption of the data buffer. The encryption key remains in the HASP HL. If the encoding operation fails, the data targeted for encryption is not affected. To decode the data buffer, use the hasp_decrypt function.

See also hasp_decrypt()

hasp_free()

Description Frees up memory resources utilized by other API functions.

Syntax

```
void HASP_CALLCONV hasp_free(char *  info)
```

Parameters

info	pointer to the memory resources to be released

Usage notes Must be used to free up memory resources allocated to store retrieved data from API calls using the hasp_get_sessioninfo() and hasp_update() functions. The function has no returns.

See also hasp_get_sessioninfo()
hasp_update()

hasp_get_rtc()

Description Reads the current time from a HASP HL Time key.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_get_rtc(
          hasp_handle_t  handle,
          hasp_time_t *  time
          )
```

Parameters

handle	handle for the session
time	pointer to the current time

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_NOT_IMPL	feature type functionality not implemented
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available
HASP_NO_BATTERY_POWER	real-time clock has run out of power
HASP_NO_TIME	real-time clock is not available

Usage notes The purpose of the function is not related to licensing, and is mainly used to obtain reliable timestamps that are independent of the system clock. The time values are returned as the number of seconds that have elapsed since Jan.-01-1970 0:00 hours UTC. Use the hasp_hasptime_to_datetime() function to convert the return into a date and time value.

See also hasp_datetime_to_hasptime()
hasp_hasptime_to_datetime()

hasp_get_sessioninfo()

Description Retrieves information regarding a session context.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_get_sessioninfo(
          hasp_handle_t  handle,
          char *  format,
          char **  info
          )
```

Parameters

handle	handle for the session
format	XML definition for the type of output data structure. There are three options: **1.** HASP_KEYINFO — format for retrieving information on the HASP HL key. **2.** HASP_SESSIONINFO — format for retrieving information on the session. **3.** HASP_UPDATEINFO — format for retrieving information on a license update usually contained in a C2V file — includes information on update counters, licenses and memory images currently available in a deployed HASP HL key.
info	pointer to the information which is retrieved as XML text

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_INV_FORMAT	unrecognized format
HASP_INSUF_MEM	out of memory

Usage notes

The retrieved session information applies to:

- A deployed HASP HL key
- The current or a specific login session
- A license update

This function allocates memory for the information it retrieves. To free up allocated memory resources, use the hasp_free function.

See also

hasp_free()

hasp_get_size()

Description Retrieves the byte size of a memory file from a HASP HL key.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_get_size(
          hasp_handle_t  handle,
          hasp_fileid_t  fileid,
          hasp_size_t *  size
          )
```

Parameters

handle	handle for the session
fileid	identifier for the file that is to be queried
size	pointer to the resulting file size

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_INV_FILEID	unrecognized file identifier
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes This function is used to determine the file size of the HASP HL memory. The file size enables you to determine the largest possible byte size offset. This information is useful when reading or writing to the memory of a key. The first byte in a file has the index 0.

See also hasp_read()
hasp_write()

hasp_hasptime_to_datetime()

Description Converts a time value (elapsed seconds since January 1 1970) into a date and time.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_hasptime_to_datetime(
          hasp_time_t  time,
          unsigned int *  day,
          unsigned int *  month,
          unsigned int *  year,
          unsigned int *  hour,
          unsigned int *  minute,
          unsigned int *  second
          )
```

Parameters

time	pointer for placing time value
day	pointer for day value
month	pointer for month value
year	pointer for year value
hour	pointer for hour value
minute	pointer for minute value
second	pointer for second value

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_TIME	time beyond the supported range

Usage note All time values are based on Coordinated Universal Time (UTC). The converted date and time value reflects the number of elapsed seconds since Jan. 1, 1970. This conversion function is used in conjunction with the API functions that set or retrieve values for the real-time clock (RTC) of a HASP HL Time.

See also hasp_datetime_to_hasptime()

hasp_legacy_decrypt()

Description Decrypts a buffer using HASP4 backward compatible encryption.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_legacy_decrypt(
          hasp_handle_t  handle,
          void *  buffer,
          hasp_size_t  length )
          )
```

Parameters

handle	handle for the session
buffer	pointer to the buffer to be decrypted
length	byte size of the buffer to be decrypted — 8 bytes minimum is required

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_TOO_SHORT	decryption data length is too short
HASP_ENC_NOT_SUPP	hardware does not support encryption type
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes Execution of this function is dependent on the handle generated during the hasp_login() call. The Feature ID must have been declared as a '`PROGRAM NUMBER FEATURE`' type. If the decryption operation does not succeed, the data pointed to by buffer remains undefined.

See also hasp_legacy_encrypt()

hasp_legacy_encrypt()

Description Encrypts a buffer using HASP4 backward compatible encryption.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_legacy_encrypt(
            hasp_handle_t  handle,
            void *  buffer,
            hasp_size_t  length
            )
```

Parameters

handle	handle for the session
buffer	pointer to the buffer to be encrypted
length	byte size of the buffer to be encrypted — 8 bytes minimum is required

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_TOO_SHORT	encryption data length is too short
HASP_ENC_NOT_SUPP	hardware does not support encryption type
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes Execution of this function is dependent on the handle generated during the hasp_login() call. The Feature ID must be declared as a '`PROGRAM NUMBER FEATURE`' type. If the encryption operation does not succeed, the data pointed to by buffer remains undefined.

See also hasp_legacy_decrypt()

hasp_legacy_set_idletime()

Description Sets the HASP License Manager (LM) idle time value.

Syntax

```
hasp_status_t HASP_CALLCONV  hasp_legacy_set_idletime(
          hasp_handle_t  handle,
          hasp_u16_t  idle_time
          )
```

Parameters

handle	handle for the session
idle_time	the idle time value in minutes

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_REQ_NOT_SUPP	attempt to set the idle time for a local key
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes Execution of this function is dependent on the handle generated during the hasp_login() call. The Feature ID must be declared as a ‘PROGRAM NUMBER FEATURE’ type. Only use this function when accessing a HASP HL key over the network.

See also hasp_login()

hasp_legacy_set_rtc()

Description Backward compatability function, setting the real-time clock in a HASP4 Time key.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_legacy_set_rtc(
          hasp_handle_t  handle,
          hasp_time_t  new_time)
          )
```

Parameters

handle	handle for the session
new time	time value to set the rtc

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available
HASP_NO_TIME	RTC not available or key access is remote

Usage notes Execution of this function is dependant on the handle generated during the hasp_login() call. The Feature ID must have been declared as a '`PROGRAM NUMBER FEATURE`' type. Use the utility functions provided to convert the time values accordingly.

See also hasp_get_rtc()
hasp_hasptime_to_datetime()
hasp_datetime_to_hasptime()

hasp_login()

Description Logs into a feature and thereby establishes a session context.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_login(
          hasp_feature_t  feature_id,
          hasp_vendor_code_t  vendor_code,
          hasp_handle_t *  handle
          )
```

Parameters

feature id	Unique identifier for a specific feature stored in a HASP HL key. The function provides the specification for the login which comprises: **1.** A '`PROGRAM NUMBER FEATURE`' type that species that the login is to a program number. **2.** A unique program number value. **3.** Specific details for the login as: • only local — only a local license • only remote — only a remote license • login is counted per process ID — per process decrementation • disable terminal server check — to run protected applications on terminal server stations • enable access to HASP3/HASP4 keys.
vendor code	pointer to the vendor code
handle	pointer to the session handle

Returns

HASP_CONTAINER_NOT_FOUND	required key not found
HASP_STATUS_OK	the request was successfully completed
HASP_FEATURE_NOT_FOUND	cannot find requested feature
HASP_FEATURE_TYPE_NOT_IMPL	the requested feature type is not available
HASP_INV_PROGNUM_OPT	unknown '`PROGRAM NUMBER FEATURE`' option requested
HASP_TMOF	too many open handles
HASP_INSUF_MEM	out of memory
HASP_INV_VCODE	invalid vendor code
HASP_NO_DRIVER	driver not installed
HASP_OLD_DRIVER	old driver installed
HASP_TS_DETECTED	program is running remotely on a Terminal Server

Usage notes

This function establishes a context to a HASP HL key containing a license for the requested program number.

The Feature ID is the corresponding '`PROGRAM NUMBER FEATURE`' that exists inside a HASP HL license container.

When the default feature, or program number '0' is used, the API only searches for the HASP HL key and not its licensing information.

The requisite vendor codes are stored in a *Vendorcodes* folder in your system. Without the correct vendor code, the function call cannot succeed. You can open up to 128 simultaneous login sessions.

See also

hasp_logout()

hasp_logout()

Description Logs out from a context or session.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_logout(
          hasp_handle_t  handle
          )
```

Parameters

handle	handle for the session being terminated

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle

Usage notes Use this function to end a connection to an API object. Once logged out from a session, all memory allocated for the session is freed. The connection to the HASP License Manager shuts down if the logged out connection was the last API session.

See also hasp_login()

hasp_read()

Description Reads the memory of a HASP HL key.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_read (
          hasp_handle_t  handle,
          hasp_fileid_t  fileid,
          hasp_size_t  offset,
          hasp_size_t  length,
          void *  buffer
          )
```

Parameters

handle	handle for the session
fileid	identifier for the file that is to be queried
offset	byte offset for the file
length	number of bytes in the file
buffer	pointer to the retrieved data

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_NOT_IMPL	the requested feature type is not available
HASP_INV_FILEID	unrecognized file identifier
HASP_MEM_RANGE	out of memory
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes

The following fileid options are available:
`HASP_FILEID_LICENSE`
`HASP_FILEID_MAIN`.
Access depends on the generated hasp_login() handle. The Feature ID must be declared as a '`PROGRAM NUMBER FEATURE`' type to restrict access to a HASP4 memory file.

Use the hasp_get_size() function to determine the size of the file you want to read.

See also

hasp_get_size()
hasp_write()

hasp_update()

Description Writes an update for a HASP HL license.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_update (
          char *  update_data,
          char **  ack_data
          )
```

Parameters

update_data	pointer to the complete update data
ack_data	pointer to a buffer to get the acknowledge data

Returns

HASP_INV_UPDATE_DATA	required XML tags not found OR contents in binary data missing or invalid
HASP_INV_UPDATE_OBJ	binary data doesn't contain an update
HASP_NO_ACK_SPACE	acknowledge data requested by the update, however, ack_data input parameter is NULL
HASP_KEYID_NOT_FOUND	HASP HL license to be updated not found
HASP_INV_UPDATE_NOTSUPP	update not supported by the HASP HL key
HASP_UNKNOWN_ALG	unknown algorithm used in V2C file
HASP_INV_UPDATE_CNTR	update counter is set incorrectly
HASP_INV_SIG	signature verification failed

Usage notes

This function writes update information.

The update code contains all necessary data to perform the update on the deployed HASP HL key including:

1. Where to write (in which “container”, e.g. HASP HL key)
2. The necessary access data - Vendor Code
3. The actual update information.

The function returns an acknowledgement code, which is signed/encrypted by the update. The code is evidence that an update has been applied to a license. Memory for the acknowledge data is allocated by the API and has to be freed using hasp_free().

This function is an extension of the main HASP HL API and is utilized by the HASP HL RUS utility to update HASP HL licenses.

See also

hasp_free()

hasp_write()

Description Writes to the memory of a HASP HL key.

Syntax

```
hasp_status_t HASP_CALLCONV hasp_write(
          hasp_handle_t  handle,
          hasp_fileid_t  fileid,
          hasp_size_t  offset,
          hasp_size_t  length,
          void *  buffer
          )
```

Parameters

handle	handle for the session
fileid	identifier for the file to write
offset	byte offset for the file
length	number of bytes in the file
buffer	pointer to the retrieved data

Returns

HASP_STATUS_OK	the request was successfully completed
HASP_INV_HND	invalid input handle
HASP_NOT_IMPL	the requested feature type is not available
HASP_INV_FILEID	unrecognized file identifier
HASP_MEM_RANGE	out of memory
HASP_ACCESS_DENIED	access to feature is denied
HASP_CONTAINER_NOT_FOUND	the feature container is no longer available

Usage notes

Valid fileids currently available for writing data:
`HASP_FILEID_MAIN`
`HASP_FILEID_LICENSE`
Access to memory files depends on the generated hasp_login() handle. When you obtain a handle after logging into any feature other than the default feature (program number 0), access is limited to HASP_FILEID_MAIN. However if you log into the default feature, you can also access HASP_FILEID_LICENSE. Be aware that writing to the HASP_FILEID_LICENSE file could destroy the existing licensing data.

See also

hasp_get_size()
hasp_read()

API Status Codes

Below is an alphabetical listing and description of possible return codes related to the operation of the HASP HL API functions.

No	Status Code	Description
0	HASP_STATUS_OK	The request was successfully completed.
1	HASP_MEM_RANGE	Your request exceeds the HASP HL memory range.
2	HASP_INV_PROGNUM_OPT	You have requested an unknown ‘PROGRAM NUMBER FEATURE’.
3	HASP_INSUF_MEM	Your system is out of memory.
4	HASP_TMOF	There are too many open handles.
5	HASP_ACCESS_DENIED	Access to feature is denied.
6	HASP_INCOMPAT_FEATURE	The legacy decryption function cannot work on the feature.
7	HASP_CONTAINER_NOT_FOUND	The HASP HL key is no longer available.
8	HASP_TOO_SHORT	Decrypted data length is too short to execute the function call.
9	HASP_INV_HND	You have passed an invalid handle to the function.
10	HASP_INV_FILEID	Specified fileid is not recognized by the API.
11	HASP_OLD_DRIVER	The installed driver is too old to execute the function.
12	HASP_NO_TIME	A real-time clock (rtc) is not available.
13	HASP_SYS_ERROR	Generic error from host system call.

No	Status Code	Description
14	HASP_NO_DRIVER	The required driver is not installed.
15	HASP_INV_FORMAT	Unrecognized file format for update.
16	HASP_REQ_NOT_SUPP	The function cannot be executed in this context.
17	HASP_INV_UPDATE_OBJ	The binary data passed to the function does not contain an update.
18	HASP_KEYID_NOT_FOUND	The HASP HL license you requested to update has not been found.
19	HASP_INV_UPDATE_DATA	The required XML tags not found. Contents in the binary data are missing or invalid.
20	HASP_INV_UPDATE_NOTSUPP	The update request is not supported by the HASP HL key.
21	HASP_INV_UPDATE_CNTR	The update counter is not set properly.
22	HASP_INV_VCODE	You have passed an invalid vendor code.
23	HASP_ENC_NOT_SUPP	The used hardware does not support the encryption type.
24	HASP_INV_TIME	The passed time value is beyond the supported value range.
25	HASP_NO_BATTERY_POWER	Real-time clock battery has run out of power.
26	HASP_NO_ACK_SPACE	Acknowledge data requested by the update ack_data parameter is NULL.

No	Status Code	Description
27	HASP_TS_DETECTED	Program is running remotely on a Terminal Server.
28	HASP_FEATURE_TYPE_NOT_IMPL	The type of feature requested is not implemented.
29	HASP_UNKNOWN_ALG	Unknown algorithm used in V2C file.
30	HASP_INV_SIG	The signature verification operation has failed.
31	HASP_FEATURE_NOT_FOUND	The requested feature is no longer available.
500	HASP_INVALID_OBJECT	Object not initialized correctly.
501	HASP_INVALID_PARAMETER	Parsed parameter is incorrect.
502	HASP_ALREADY_LOGGED_IN	Logging in twice to the same object.
503	HASP_ALREADY_LOGGED_OUT	Logging out twice from the same object.
525	HASP_OPERATION_FAILED	Error related to incorrect use of system or platform.
698	HASP_NOT_IMPL	The requested type of feature is not implemented.

Appendix D
HASP HL Hardware Technical Specifications

Table D.1 HASP HL hardware Technical Specifications

Specification	HASP HL 'large'	HASP HL 'mini'
Model types	HASP HL Time, HASP HL Net, HASP HL Max, HASP HL Pro, HASP HL Basic and Master HASP HL	HASP HL Net, HASP HL Max, HASP HL Pro and HASP HL Basic
Length	52 mm	38 mm
Height	8 mm	8 mm
Width	16 mm	16 mm
Weight	5.4 g 6.1 g for HASP HL Time	4.4 g
Connector	USB type A	USB type A
Lines used	Power, ground, 2 for data	Power, ground, 2 for data
Plastic case material	Lexan 500R	Lexan 500R
Operating temperature	-25°C to 70°C -13°F to 158°F	-25°C to 70°C -13°F to 158°F

Specification	HASP HL 'large'	HASP HL 'mini'
Data retention	10 years minimum	10 years minimum
Memory write cycles	1.000.000	1.000.000
LED color	red	red
Humidity rating	0-100% without condensation	0-100% without condensation
Power consumption	Operating: 50 mA Standby: <0.5 mA	Operating: 50 mA Standby: <0.5 mA

Table D.2 HASP HL Model Technical Specifications

Specification	HASP HL Basic	HASP HL Pro	HASP HL Max	HASP HL Net	HASP HL Time
Memory size (bytes)	-	112	4032	4032	4032
Battery lifetime	none	none	none	none	4 years

Index

C

D

E

F

G

H

I

L

M

N

O

P

X

Aladdin®
SECURING THE GLOBAL VILLAGE

For more info: **eAladdin.com/HASP**

North America	T: 1-800-562-2543, 1-847-818-3800, F: 1-847-818-3810, Email: HASP.us@eAladdin.com
International	T: +972-3-636-2222, F: +972-3-537-5796, Email: HASP@eAladdin.com
UK	T: +44-1753-622266, F: +44-1753-622262, Email: HASP.uk@eAladdin.com
Germany	T: +49-89-89-42-21-0, F: +49-89-89-42-21-40, Email: HASP.de@eAladdin.com
Benelux	T: +31-30-688-0800, F: +31-30-688-0700, Email: HASP.nl@eAladdin.com
France	T: +33-1-41-37-70-30, F: +33-1-41-37-70-39, Email: HASP.fr@eAladdin.com
Spain	T: +34-91-375-99-00, F: +34-91-754-26-71, Email: HASP.es@eAladdin.com
Israel	T: +972-3-636-2222, F: +972-3-537-5796, Email: HASP.il@eAladdin.com
Asia Pacific	T: +852-2166-8605, F: +852-2166-8999, Email: HASP.as@eAladdin.com
Japan	T: +81-426-60-7191, F: +81-426-60-7194, Email: HASP.jp@eAladdin.com

05772

Printed in Israel