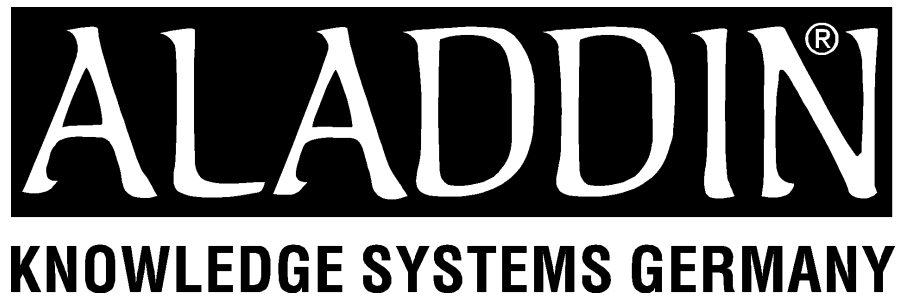


Hardlock Server Installation API

Application Programming Interface

© 1998



Aladdin Document: HSI API
Revision: 1.01e
Date: Oct 19, 1998

1 Contents

1	Contents.....	1-2
2	Introduction	2-3
3	Installation API Functions	3-4
3.1	WORD HSIGetVersion (LPTSTR Path).....	3-5
3.2	HSIRET HSIGetStatus (void)	3-6
3.3	HSIRET HSIInstallServer (DOWRD Flag, LPTSTR src, LPTSTR dest)	3-7
3.4	HSIRET HSIStartServer (LPTSTR Name)	3-8
3.5	HSIRET HSIStopServer (void)	3-9
3.6	HSIRET HSIRemoveServer (DWORD Flag)	3-10
3.7	HSIRET HSIAddHardlock (WORD Modad)	3-11
3.8	HSIRET HSIRemoveHardlock (WORD Modad)	3-12
3.9	HSIRET HSISetTimeout (LONG Timeout)	3-13
3.10	HSIRET HSIEnableProtocol (WORD Protocol)	3-14
3.11	HSIRET HSIDisableProtocol (WORD Protocol)	3-15
3.12	LPTSTR HSIGetLastErrorText (LPTSTR Text, LONG size).....	3-16
3.13	HSIRET HSIGetOSVersion (void).....	3-17
4	Appendix.....	4-18
4.1	Overview of HL-Server Installation API Functions.....	4-18
4.2	Table of API return codes.....	4-19
4.3	Table of API option flags.....	4-20

2 Introduction

The Hardlock Server Installation API provides you with all necessary functions to include the installation process into your application.

Hardlock Server for Windows NT has to be installed as a service on the system. Hardlock Server for Windows 9x is an application that needs no further installation except a copy of the file itself.

All these tasks can be automated using the Hardlock Server Installation API (HSI API). It also provides you with function to configure server settings by your application.

The HSI API is implemented as Win32 functions, which can be used in a DLL to be used dynamically (HSIAPID.DLL) or as a library which can be linked statically to your application (HSIAPIL.LIB).

3 Installation API Functions

In the following section we have provided you with a description of the various functions. All program examples are in C code.

On this page you will find an explanation of how the individual high-level function descriptions are structured.

Structure of Function Descriptions

FunctionName ([Argument1[,Argument2[,...]]])

Purpose: Brief description of parameter

Arguments: List of arguments that can be used with this parameter.

Output: List of all values that this function outputs to the application (for example, error messages).

Use: Detailed description of how the parameter works, information on how it is used.

Uses: List of all low-level API routines addressed (used) by the high-level function.

Example:

The examples in this section are written in C code and are only designed to roughly demonstrate how the function is used.

3.1 WORD HSIGetVersion (LPTSTR Path)

Purpose: Returns the version of the currently installed HL-Server.

Arguments: Pointer to path or NULL

Output: The function directly returns the version of the service as follows:
High Byte: Major Version decimal
Low Byte: Minor Version decimal

Use: This function gets the version of the installed service under Windows NT and the version of running server under Windows 9x if the given parameter is NULL. As an alternative it can point to the path to the file where to get the version from.

Note: the version function is only supported since HL-Server 3.30, older implementations return 0 as version.

Example: Both calls in the example return the same value if called under Windows NT.

```
WORD ver;
```

```
Ver = HSIGetVersion(NULL);  
printf("Server version is: %2d.%02d", Ver >> 8, Ver & 255);  
....  
Ver = HSIGetVersion("f:\\winnt\\system32\\hls32svc.exe");  
printf("Server version is: %2d.%02d", Ver >> 8, Ver & 255);
```

3.2 HSIRET HSIGetStatus (void)

Purpose: Retrieves the current status of the HL-Server.

Arguments: (None)

Output: The function may return the following values:

HSI_SERVER_NOT_INSTALLED:

The server service is not installed on this system.

HSI_SERVER_NOT_RUNNING:

Windows NT: the server service is installed but not started.

Windows 9x: the server is not started.

HSI_SERVER_RUNNING:

The server is installed and started.

Use: Get status of service.

Example:

```
rc = HSIGetStatus();
switch (rc)
{
case HSI_SERVER_NOT_INSTALLED:
    printf("Server Service is not installed.");
    break;
case HSI_SERVER_NOT_RUNNING:
    printf("Server service installed but not running.");
    break;
case HSI_SERVER_RUNNING:
    printf("Server service installed and running.");
    break;
}
```

3.3 HSIRET HSIInstallServer (DOWRD Flag, LPTSTR src, LPTSTR dest)

Purpose: Installs the HL-Server service.

Arguments: Pointer to string containing the source path, pointer to string containing the destination path.
See appendix for supported flags.

Output: see appendix for possible API return codes.

Use: This function installs the Hardlock server service. If the first parameter is NULL, the EXE (HLS32SVC.EXE for Windows NT, HLS32.EXE for Windows 9x) is taken from the current directory. If you want take the file from another directory, you have to specify the source path (including the file name) as first parameter. By default the service is installed in the Windows directory (e.g. Windows NT: F:\winnt\system32\, Windows 9x: F:\win95\system\). For Windows 9x you can specify a destination with the second parameter, it is ignored under Windows NT.

With this function the server/service will only be installed. You have to use HSIStartServer() to start the server.

Example:

```
HSIRET rc;  
  
rc = HSIInstallService(0, NULL, NULL);  
if (rc == HSI_FAIL)  
    printf("Cannot install HL-Server.");
```

3.4 HSIRET HSISStartServer (LPTSTR Name)

Purpose: Starts the installed HL-Server.

Arguments: Filename of Application to start.

Output: see appendix for possible API return codes.

Use: Start the server. The parameter is ignored under Windows NT and the server service is started. Under Windows 9x you can specify the name and path of the application to start. If the parameter is NULL, the function tries to start HLS32.EXE from the search path.

Example:

```
rc = HSISStartServer("c:\\hardlock\\hls32.exe");  
if (rc == HSI_FAIL)  
    printf("Cannot start HL-Server.");
```

3.5 HSIRET HSISStopServer (void)

Purpose: Stops the installed HL-Server.

Arguments: (None)

Output: see appendix for possible API return codes.

Use: Stops the server

Example:

```
rc = HSISStopService();  
if (rc == HSI_FAIL)  
    printf("Cannot stop HL-Server.");
```

3.6 HSIRET HSIRemoveServer (DWORD Flag)

Purpose: Stops and removes the server from the system.

Arguments: See appendix for supported flags.

Output: see appendix for possible API return codes.

Use: Remove the server.

Example:

```
rc = HSIRemoveService(0);  
if (rc == HSI_FAIL)  
    printf("Cannot remove HL-Server.");
```

3.7 HSIRET HSIAddHardlock (WORD Modad)

Purpose: Adds a Hardlock to be served by the server.

Arguments: Hardlock Module Address to be served.

Output: see appendix for possible API return codes.

Use: The function adds a Hardlock with the specified module address to be served by the service. This function is normally not needed, since HL-Server support "AutoAdd" since version 3.30. The means if a application sends a request to the server, the server itself checks for the correct Hardlock's and adds it the list of Hardlock's to be served.

Example:

```
rc = HSIAddHardlock(29809);  
if (rc == HSI_FAIL)  
    printf("Cannot add Hardlock with module address 29809.");
```

3.8 HSIRET HSIRemoveHardlock (WORD Modad)

Purpose: Removes a Hardlock to be no longer served by the service.

Arguments: Hardlock Module Address to be removed.

Output: see appendix for possible API return codes.

Use: The function removes a Hardlock with the specified module address to be served by the service.

Example:

```
rc = HSIRemoveHardlock (29809);  
if (rc == HSI_SUCCESS)  
    printf("Hardlock 29809 now removed.");
```

3.9 HSIRET HSISetTimeout (LONG Timeout)

Purpose: Change the timeout for HL-Server.

Arguments: Timeout in minutes.

Output: see appendix for possible API return codes.

Use: You can change the default timeout (15 minutes) after a login is freed if no request was submitted by the application. Please refer to the Hardlock Server manual for further information.

Example:

```
rc = HSISetTimeout(20);
if (rc == HSI_SUCCESS)
    printf("Timeout now 20 minutes.");
```

3.10 HSIRET HSIEnableProtocol (WORD Protocol)

Purpose: Enable the protocols the service will handle.

Arguments: Protocols as flags. Supported defines are IP, IPX and NETBIOS.

Output: see appendix for possible API return codes.

Use: By default HL-Server serves all protocols found on the system. After you disabled a protocol you are able to enable it again using this function. Please refer to the Hardlock Server manual for further information.

Example:

```
rc = HSIEnableProtocol(IP | IPX);  
if (rc == HSI_SUCCESS)  
    printf("IP and IPX enabled.");
```

3.11 HSIRET HSIDisableProtocol (WORD Protocol)

Purpose: Disables the protocols the service will handle.

Arguments: Protocols as flags. Supported defines are IP, IPX and NETBIOS.

Output: see appendix for possible API return codes.

Use: By default HL-Server serves all protocols found on the system. You are able to disable a specific protocol using this function. Please refer to the Hardlock Server manual for further information.

Example:

```
rc = HSIDisableProtocol(IPX);  
if (rc == HSI_SUCCESS)  
    printf("IPX now enabled.");
```

3.12 LPTSTR HSIGetLastErrorText (LPTSTR Text, LONG size)

Purpose: Retrieves the text of the last error occurred.

Arguments: Pointer to buffer to receive the error text, size of buffer.

Output: Pointer to buffer to receive the error text.

Use: This function uses the systems GetLastError() and returns the message text of the last error that occurred.

Example:

```
Char * MyText[256];  
  
HSIGetLastErrorText(MyText, 256);  
printf("LastError: %s", MyText);
```

3.13 HSIRET HSIGetOSVersion (void)

Purpose: Retrieves the current operating system.

Arguments: (None)

Output: The function returns the following values:

WIN_32s:
Running on Windows 3.x with Win32s installed.

WIN_9x:
Running on Windows 95 or 98.

WIN_NT:
Running on Windows NT.

Use: Get version of operating system. The HSI API will only run on Windows 95/98 and Windows NT.

Example:

```
rc = HSIGetOSVersion();
switch (rc)
{
    case WIN_32s:
        printf("Running on Win32s.");
        break;
    case WIN_9x:
        printf("Running on Win9x.");
        break;
    case WIN_NT:
        printf("Running on WinNT.");
        break;
}
```

4 Appendix

4.1 Overview of HL-Server Installation API Functions

The chart on the following pages provides you with an overview of all API functions.

High-level Function	Use
HSIGetVersion (LPTSTR)	Return version of installed service.
HSIGetStatus ()	Get status information about the service.
HSIInstallServer (DWORD, LPTSTR, LPTSTR)	Install Server.
HSIStartServer ()	Start service.
HSIStopServer ()	Stop service.
HSIRemoveService ()	Remove service.
HSIAddHardlock (WORD)	Add Hardlock to be served.
HSIRemoveHardlock (WORD)	Remove Hardlock.
HSISetTimeout (LONG)	Set timeout for HL-Server.
HSIEnableProtocol (WORD)	Enable a specific protocol.
HSIDisableProtocol (WORD)	Disable a specific protocol.
HSIGetLastErrorText (LPTSTR, LONG)	Retrieves the error text from the last system call (uses GetLastError()).
HSIGetOSVersion()	Get version of operating system.

4.2 Table of API return codes

No.	Designation	Meaning
0	HSI_SUCCES	Function succesful.
1	HSI_FAIL	Function failed.
2	HSI_SERVER_NOT_INSTALLED	The service is not installed.
3	HSI_SERVER_NOT_RUNNING	The service is installed but not started.
4	HSI_SERVER_RUNNING	The service is installed and started.
5	HSI_OPENSERVICE_FAILED	Failed to open service, probably not installed.
6	HSI_OPENSCMANAGER_FAILED	Failed to call the service manager.
7	HSI_CREATESERVICE_FAILED	Service cannot be created.
8	HSI_FILECOPY_FAILED	Copy of HLS33SVC.EXE failed, probably file not found in the specified directory.
9	HSI_OS_NOT_SUPPORTED	Functionnot supported under the current operating system

4.3 Table of API option flags.

Designation	Meaning
HSI_DO_NOT_DELETE_FILES	Do not delete any files during driver remove.