LispWorks® for UNIX

Guide to the License Server

Version 7.1

Copyright and Trademarks

LispWorks Guide to the License Server

Version 7.1

September 2017

Address

LispWorks Ltd
St. John’s Innovation Centre
Cowley Road
Cambridge
CB4 0WS
England

Telephone

From North America: 877 759 8839 (toll-free)

From elsewhere: +44 1223 421860

Fax

From North America: 617 812 8283

From elsewhere: +44 870 2206189

www.lispworks.com

Contents

About this Guide

Introduction

This guide describes the License Server, **hqn_lsd**. The License Server is an optional complement to the static keyfile system described in the *LispWorks Release Notes and Installation Guide*.

Unless you intend to run Lisp in a way that causes problems for the keyfile system (for instance, licenses floating across a network), you are unlikely to need to use the License Server. If you intend to use the keyfile system, you can ignore the rest of this manual.

Note that it is possible to configure Lisp to take certain permissions from a keyfile and the rest from the License Server. This might be useful if, for example, you have floating licenses for the Lisp image but a fixed license for a layered add-on product such as CLIM.

Topics covered by the Guide

Chapter 1, “The License Server”, describes how to install and run the License Server in conjunction with Lisp. In many cases, the information in this chapter may be all you need to get LispWorks products running with the License Server.

Chapter 2, “The License Server Daemon”, describes **hqn_lsd**, the License Server daemon. This chapter provides information additional to Chapter 1 that may be required in certain circumstances.

Chapter 3, “The License Server Admin Tool” describes **`hqn_lsa`**, the License Server administration tool. This tool lets you configure the License Server to your site, and allows finer control over the way licenses on your network or machine are managed.

1

The License Server

1.1 Introduction

This manual describes how to set up and use the License Server, a network license server for LispWorks and related LispWorks products. It covers:

- How to use the License Server daemon, **`hqn_lsd`**.
- How to use the License Server administration tool, **`hqn_lsa`**.

The License Server daemon (**`hqn_lsd`**) is designed to provide secure network licensing services for any number of application products.

The License Server can be used instead of the keyfile system.

1.2 Installing Lisp for use with the License Server

This section describes changes to the installation procedure for LispWorks products to allow for use of the License Server daemon.

The installation of the server daemon itself is explained in Chapter 2, “The License Server Daemon”.

Invocation of License Server from Lisp is controlled by the environment variable **LW_CHECK_NETWORK**. If this is not set in your environment, then the Lisp variable **system:*check-network-server*** is used instead. See Section 1.4 for full details of the treatment of these two variables.

Instructions for performing a typical, "quick" installation:

1. Set the environment variable **LW_CHECK_NETWORK** to any value. (The test is of whether the variable has been set or not.)
2. Make a configuration file wherein the variable ***check-network-server*** is set:

   For example for a release (say, 7.1) of LispWorks you would make a copy of **/path/to/lib/7-1-0-0/config/configure.lisp** called **my-configuration.lisp** and add the form

   ```
   (setq system:*check-network-server* t)
   ```

   in **my-configuration.lisp**.
3. Configure Lisp as usual, following the instructions in the *LispWorks Release Notes and Installation Guide*.

Setting **LW_CHECK_NETWORK** allows you to start Lisp the first time, and saving non-nil ***check-network-server*** into the configured image allows you to restart it in the future.

1.3 Using the License Server

The Lisp distribution contains three executables: **hqn_lsd**, **hqn_lsa**, and **hqn_cp**. They may have platform-specific names in your distribution (for example, **hqn_lsa-rs6k**), but throughout this manual we use the generic names. The UNIX manual pages for each of them are the documentation tar file on your distribution CD—see the *Release and Installation Notes* for more precise details.

These executable files should be installed on your system in an appropriate place (for instance, **/usr/local/bin**). Permits for the License Server will be supplied to you by LispWorks Ltd. Also see Section 2.9 on page 16 for information on **hqn_cp**.

The basic procedure for running the License Server with LispWorks and its add-on products consists of four steps:

1. Add machine-name aliases **hqnserver1**, **hqnserver2**, ..., **hqnserver***n* to each of the *n* machines to be used for running the License Server.
2. Place the License Server permits in a suitable directory on each of these *n* machines. If a permit is placed in the same directory as the executable, then that License Server should run without further configuration.
3. Run **hqn_lsd** on each of these machines, and edit your **rc.local** files (or whichever files are appropriate for your systems) so that **hqn_lsd** is automatically started whenever the machines are rebooted.
4. Reconfigure the Lisp image according to the instructions in Section 1.2, "Installing Lisp for use with the License Server", above.

For full details on the first two steps, you should refer to Section 2.3, "Setting up domains with more than one host" and Section 2.5, "Location of license permit files".

1.4 Interaction between Lisp and hqn_lsd

Note that the following description applies equally to the Lisp image and to any layered products.

Each time you start your image (or load a layered product, like CLIM, into an already running image) a check is made to ensure that you do have a valid license for the product in question. Depending on the value of the UNIX environment variable **LW_CHECK_NETWORK**, or the Lisp variable **sys:*check-network-server***, the system will check for standalone keys (that is, in keyfiles); if any keyfile check fails and network licensing has been enabled, then the system will request a license from **hqn_lsd**.

LW_CHECK_NETWORK can take any of the following values, which are case-insensitive:

No	Do not attempt to contact the License Server.
Only	Only try the License Server; do not try the keyfile.

Any other value

Try keyfiles first and then the Licence Server, if necessary.

The main use of the environment variable is to allow you to configure your image without reading local configuration files into Lisp first, which would require the extra step of requesting a keyfile from LispWorks Ltd.

If **LW_CHECK_NETWORK** is not set, the action is controlled by the value of **sys:*check-network-server***. This, in turn, can take the following values:

nil Do not attempt to contact License Server.

:only Only try the License Server; do not try the keyfile.

Any other non-**nil** value

Try keyfiles first and then the License Server if necessary.

The number of licenses issued by LispWorks Ltd to each customer site corresponds to the number of Lisp processes which may be simultaneously active at that site. As a general rule, they may be run on any machines at that site. This differs from the standalone system in two ways:

- Under the standalone system, Lisp processes may only be run on licensed machines
- Under the standalone system, any number of processes may run on each licensed machine (resources permitting)

Once a license has been granted to a running Lisp image by **hqn_lsd** it remains assigned to that UNIX process until the process terminates. If Lisp terminates normally—for instance by calling **(lw:quit)** or the editor command **Save All Files And Exit**—then the license will be released for reuse, automatically and immediately.

On the other hand, if Lisp terminates abnormally (for example, it was killed externally) then the license will not be released upon termination. However the License Server will notice within 10 minutes that the Lisp process has stopped confirming its license, and at this point the daemon itself will release the license.

The administration tool, **hqn_lsa** (see Chapter 3) can be used at any time to release any license. If this license belongs to a process that has terminated abnormally then it can now be reused. If the license belongs to an active Lisp process then that process will request a new license from the server daemon.

If a running Lisp image is stopped (with **Ctrl+Z**) or spends a very long time garbage collecting, then the License Server may detect that the image is no longer active and release its license. In the normal course of events, the image will automatically request a new license from the server daemon as soon as it becomes active again.

If, for whatever reason, a running image requests a new license because its old one cannot be found (for example, it has been released externally, or the License Server and its backups cannot be contacted) then

- If the request is successful, the user will not be informed that the transaction ever took place.
- If the request is unsuccessful, a warning message will be written to ***terminal-io***, and the image will continue. The image will automatically make regular attempts to regain the license.

  A typical example:

  ```
  ;; *** License for LispWorks lost - access to this software
  ;; disabled.
  ;; *** LispWorks is regularly checking for a license to become
  ;; available.
  ```

  This may be followed later by the message

  ```
  ;; *** License for LispWorks recovered.
  ```

- If the request is unsuccessful because all possible licenses have been assigned to other users, then a list of active users will be printed.

2

The License Server Daemon

2.1 Overview

The License Server daemon, **`hqn_lsd`**, provides secure network licensing services for Lisp. Together with its administration utility, **`hqn_lsa`**, it allows flexible but simple centralized control over the use of licensed applications throughout a network.

The daemon itself is licensed to run on one or more specific host machines at any given site. Where multiple hosts are specified this constitutes a *server domain*. Any host within a server domain can be configured as the active license server, while the rest provide backup services in case of failure of the current server.

Product licenses are provided in human-readable permit files.

Product licenses are locked to a particular server domain, that is, only daemons within that domain will serve licenses for the product (the product itself can of course be run anywhere within the network depending on its policy configuration).

2.2 Using hqn_lsd

hqn_lsd can be run on any machine, either stand-alone or within a network. On start-up, it looks for permit files in one or more defined directories. See Section 2.5, “Location of license permit files”.

Each permit file contains a list of machine-specific details for one or more machines. This list constitutes the server domain of the product permits defined in the rest of the permit file. Machines are specified by a unique ID number (the source of which depends on the kind of machine) and an Internet address. The rest of the permit file contains permit details for one or more products.

If **hqn_lsd** reads a permit file in which the server-domain listing contains the host it is running on, then it is enabled to serve licenses for the products to which the permit relates. Whether it *will* serve licenses for those products depends on the type of the server domain.

If the server domain contains a single host, **hqn_lsd** will always serve licenses for products having that domain. In this case, if the product is to be run *solely* on this host, nothing else needs to be done: just start **hqn_lsd** on this host and the licensed applications can be run (depending on the permit details of each product). If the product is to be run on other machines the server host needs an alias as described in Section 2.3 on page 8.

The case of multiple-host server domains is described below.

2.3 Setting up domains with more than one host

If the server domain contains multiple hosts, an order must be imposed on the hosts, so that one can be considered the primary license server for the domain, and the remainder can be considered backups. This ordering is achieved by defining alias hostnames for the hosts of the server domain in **/etc/hosts**.

All the hosts in the server domain should have an alias of the form

hqnserver*n*

where *n* is 1, 2, and so on. The lowest-numbered **hqnserver** alias becomes the primary license server within the server domain, and the rest become backups that will be interrogated in ascending order.

Note: It is perfectly possible to have a host occur in more than one server domain. Thus a host might be the primary license server in one server domain, and a backup in another.

To prevent product permits being duplicated on subsets of a server domain (when a network is divided, for example), **`hqn_lsd`** uses a validation mechanism in domains with multiple hosts. Upon start-up, the primary license server within a server domain must contact the daemons on at least half the members of the domain before it will serve licenses, and backup daemons must contact the acting primary before they will serve licenses in the future. This means that on initial start-up at least half of the daemons in a server domain must be active, but once an active primary is available, daemons within the domain can be started and stopped as required, primary-status always passing to the lowest-numbered **`hqnserver`** machine.

If multiple hosts are required there should ideally be a minimum of three within the server domain. This allows one machine to be down or removed from a network while still allowing the server domain to be initialized. The larger the domain, the greater the flexibility, but the longer it will take to initialize. The maximum number of hosts permitted in a server-domain is twenty.

This product/server domain mechanism allows licensed stand-alone hosts to be included in networks, or networks to be split, without new permits being required and without compromising the licensing security.

Daemons can be run on any machine, but they serve licenses only when their host machine is within the server domain of a product permit. They will, however, read all available permit files, and redirect any requests for licenses to an appropriate domain.

2.4 Product permits

Permits for application products determine the use of that application at any particular client site. Permits are human-readable text files with embedded lock strings. Permits can be altered only if the correct lock string is provided. Permit files can be obtained from LispWorks Ltd upon purchase of your Lisp product. Figure 2.1 shows a sample permit file.

```
############ The license server permit ##############

THE HARLEQUIN LICENSE SERVER VERSION 2.0

NUMBER: 680

# Licensed to run on:

SERVERS: 1
# Internet-address      Unique-ID
  192.124.144.205       0x6902b805

PERMITS: 1

#-----------------------------------------------------------#
   SXMi35wKZTM48C6FX3@Z2<W2?         g?PAu5AYfXJ2QXMP?5SOb=94O
#-----------------------------------------------------------#

Harlequin Limited
Liquid
Liquid

# Licence Node     Floating  Start      Expiry     Update
# Policy  Licences Licences  Date(dmy)  Date(dmy)  Period(min)
  0x41    0        5         19/08/1998 19/08/2010 5

# Application data (signed values)

DATA: 4

      3 0 0 0

LOCKED-NODES: 0

#-----------------------------------------------------------#
   fOonr3ToY?<3p3mwJ576=ts5?         qCno@5ov@:25Ze5Z32rUStK5?
#-----------------------------------------------------------#
```

Figure 2.1 Sample permit file

Each permit contains information about:

1. The product version.
2. The licensing policy associated with the product.

3. The number and types of licenses available.
4. Start and expiration dates of the permit.
5. Data specific to the application. For example, features that are enabled by the permit, or extra security checks.
6. If node-locked licenses are specified, details of hosts to which the product is locked.

Two basic types of license are provided: floating and node-locked licenses. If ten floating licenses are specified in a product permit then up to ten instances of the product may be run simultaneously anywhere on the network (assuming access to the current license server for the associated server domain). Node-locked licenses allow the product to be run only on certain hosts. These hosts may be specified in the product permit or, if the permit allows it, they may be configured by your systems administrator. Depending on the policy settings for the permit, floating licenses may be converted into node-locked licenses according to your requirements.

2.5 Location of license permit files

When **hqn_lsd** starts up, or is reinitialized via **hqn_lsa -reinit** (see Chapter 3, “The License Server Admin Tool”, for details of this option), it looks for permit files that match the regular expression

```
.*hqn.permit[0-9]*$
```

That is, any string, followed by the string **hqn.permit**, and possibly followed by a number. For example, **hqn.permit**, **hqn.permit1**, **Lisp-hqn.permit**, all match this pattern.

The directories that **hqn_lsd** searches for permits can be specified by one of the following methods.

The first is to supply the **-p** option on **hqn_lsd**’s command line. This option takes a colon-separated list of pathnames to search. If you do not use the **-p** option, or no permit files are found in the directories specified using it, **hqn_lsd** consults value of the environment variable **HQNPERMITPATH**, also a colon-separated list of search paths. If the variable has no value, or no permit files are found using it, the following directories are searched in order:

1. **hqn_lsd**'s current working directory
2. **/usr/local/HQN**
3. **/usr/local**

The order in which the permit files are read is determined by their order within the directory (that is the order shown by **ls -f**).

hqn_lsa -sinfo shows what permit files have been read and in what order. If no permits are found, **hqn_lsd** remains idle until given further instructions, for instance by using **hqn_lsa -reinit** to re-initialize it.

For any permit file for which **hqn_lsd**'s host is its entire server domain, **hqn_lsd** will now serve licenses for the products listed in the permit file. If the products are to be run on other machines, the server host requires an **hqnserver1** alias. For multiple-host server domains (see Section 2.3), at least $n/2$ (rounded up) daemons must be started before any licenses will be served, where n is the number of hosts in the server domain.

If any errors are detected in a permit file (such as an incorrect lock string), **hqn_lsd** discards the rest of the file and issues error messages to the system log and standard error. The cause and position of the error can also be found with **hqn_lsa -sinfo**.

2.6 Policies

Each product permit has a number of settings that determine how the product can be used. Generally the policy setting can be adjusted by your systems administrator using the administration tool **hqn_lsa**. See Chapter 3, "The License Server Admin Tool", for details of how to change policies. However, if required, LispWorks Ltd can lock one or more policy flags into the permit.

The policy value in a permit is a set of bit flags which, if set, turn on the policies shown in Table 2.1. Note that **`hqn_lsa`** never changes the permit files themselves; it only configures an active **`hqn_lsd`** daemon. Current policies are as follows for the 'on' setting.

Bit	Policy	Description
0	GRAB	Licenses not updated within 'crashout' time period are freed on next request (the default setting). Thus licenses held by applications that have crashed will be made available. This policy is on by default.
1	ONPN	Only 1 floating license is allowed per machine. For locked nodes only one license is allowed per *listed machine* — the same machine may be listed more than once to allow up to a maximum number of products instances to run on a single specified machine.
2	TIML	Licenses may only be held continuously for some maximum time period (timeout) after which the license may be given up to another user if no free licenses are available.
3	LKNL	As for TIML but the time limit does not apply to locked nodes.
4	LKNF	Node-locked machines are not allowed to also hold floating licenses.
5	LKPR	Users on node-locked machines have priority over users on other machines. That is, if no licenses are available the oldest floating license not running on a locked-node will be reallocated.
6	UVIS	The application is allowed to request a list of current users of that product so that users can see where licenses are allocated.

Table 2.1 Policies for permits.

Bit	Policy	Description
7	USRL	The system administrator can allocate guaranteed licenses for particular users. If no licenses are available the oldest current license is reallocated to the privileged user.
8	ACCL	Your system administrator can assign an access level to groups and/or particular users. Generally full access is assigned unless the current user or their current group has a lower access level assigned. Access levels are in the range 0 to 8, where 8 is full access, 1 is minimum access, and 0 denies all access to the software, that is, no license is issued.

Table 2.1 Policies for permits. (Continued)

In addition to tailoring the licensing policy of a product to your site, **hqn_lsa** can also be used to convert floating licenses to node-locked licenses (and vice versa but only if the license has been previously converted), add to the list of node-locked hosts, and exclude certain hosts from running the product at all.

2.7 Versioning

By default **hqn_lsd** will only serve licenses for one version of a product at a time. If **hqn_lsd** reads two permits for the same product, which differ in their version string, only the first permit read will be served. This is the default mechanism since it is generally required that new versions supersede old versions without increasing the total set of licenses for the product.

2.8 Logging

If required, all license transactions can be logged via the standard **syslog** daemon services. Details of product, license, user and host are logged as a **LOG_USER** facility at the **LOG_INFO** level for each basic transaction, and at the **LOG_WARNING** level for certain error conditions. If this is required, configure the **syslog.conf** file accordingly. The type and extent of logged information is configurable with the **-loglevel** option to **hqn_lsa**.

2.9 Changing permit details

The details of a permit are locked by the encrypted lock string pairs which occur at the end of each product permit. Permits can be changed by editing the permit file and obtaining a new 25-character lock string from LispWorks Ltd. The utility script **`hqn_cp`** is provided to facilitate this. See its UNIX manual page for full details.

3

The License Server Admin Tool

3.1 Overview

The **hqn_lsa** utility is available to the superuser for interrogating and configuring the **hqn_lsd** License Server daemon. It accepts commands either from the command line, from an interactive shell, from the standard input or from a file. The commands available are the same as the command line options, but the leading hyphen is omitted. It is called from a UNIX shell as follows:

hqn_lsa [-*option*][- | *filename*]

If no options are given **hqn_lsa** enters a simple interactive mode, with a prompt which indicates the currently connected server(s). By default it attempts to connect to the lowest numbered active **hqnserver** on startup, or if none are available, to the local host. The **-w** option can be used to adjust the maximum time it will try to make a connection.

With '**-**' or *filename* commands are read from **stdin** or a file, and a prompt is not supplied. The **-y** option can be used to suppress confirmations required by certain commands.

3.2 Available options and commands

All commands have a long and a short form. Values in **[]** are optional. When used on the command line, prepend a '**-**' to the command. All numerical values can be input in decimal, octal (beginning at **0**) or hexadecimal (beginning at **0x**).

Product permits can be referred to either by name or by index.The variable *p* in the options described below can be either a regular expression, which is matched against the concatenated product and version strings of all product permits held on the connected server(s), or an index number as returned by the **listp** command.

server *Command*

Syntax

s | server [*host* | *n* [*host* | *n* ...]]

On start up **hqn_lsa** attempts to connect to the **hqn_lsd** server on the host aliased to **hqnserver1**. If this fails it tries **hqnserver2** and so on. If no **hqnserver** can be contacted it attempts to connect to a server on the local host. **server** without options connects to this default server. Otherwise it attempts to connect to a server on *host* or to **hqnserver** *n*. In interactive mode the prompt indicates the connected server. Multiple hosts / indices can be given to connect to multiple hosts simultaneously, in which case all commands are copied to each connected server. The prompt indicates the number of connected servers.

listhqn *Command*

Syntax

hq | listhqn

List the hosts with hqnserver aliases.

sinfo *Command*

Syntax

i | sinfo

Gives information on the currently connected server(s). The following information is shown:

- The version string of the server
- The number of **hqnserver** aliases found by this host
- The number of products for which it is serving licenses
- The number of products seen but which are not in the domain of this host
- List of permit files read with versions and permit IDs
- List of all server-domains stored with current state

If any error occurred when reading a permit file, the error condition and number appear beside the relevant file name. Use **-reinit** to force the server to reread all permit files.

listp *Command*

Syntax

lp | listp [*p*]

Lists information on all products served or product(s) *p*. If *p* is not given, a short listing is given for each product held, showing the following information:

- Index of the product as held by the server
- File from which the permit was read
- Publisher, Product and Version descriptor strings

If *p* is given, more information is listed for each product matching *p* on each of the currently connected servers:

- Current policies in effect

- Node-locked licenses remaining out of total available
- Floating licenses remaining out of total available
- Crashout time (minutes) — the period the server will wait before deciding an application has crashed and releasing its license
- Timeout period (minutes) — If **TIMEOUT** policy is set the maximum consecutive period an application can hold a license
- Start date of product permit
- Expiry date of product permit
- Status of this host in its domain
- Server-domain of this product.
- Node locked hosts (if any)
- Excluded hosts (if any)
- Users with user-locked licenses (if any). Only if policy on
- Users with access levels set (if any). Only if policy on
- Groups with access levels set (if any). Only if policy on

If the product(s) *p* is not served by the current server then only the domain information is available.

listp+ *Command*

Syntax

lp+ | **listp+**

Give full listing on all products.

listu *Command*

Syntax

u | **listu** ***p***

Show information on current license holders of product(s) *p*. Lists the following information:

- The license index (between 1 and the total number of licenses available)
- The user name of the holder
- The host running the product application
- The type of license (locked or floating)
- The access level of the license holder
- The process ID of the application
- The time and date the license was allocated

help *Command*

Syntax

h | help

Lists all options available.

The following commands configure the connected server(s).

policy *Command*

Syntax

p | policy [*p* [*val* | *mnemonic*[-|!] ...]]

Set the policy flags for product(s) *p*. See Section 3.4 on page 27 for a description of the policy flags. If a value *val* is given this replaces the current policy value (except where policies are locked by the permit). If one or more flag mnemonics are given they turn on the corresponding policy. Appending a '**-**' turns off the policy. Appending a '**!**' locks the policy (i.e. it will not be possible to change it without re-reading the permit file). With no options, this command lists the possible policies and mnemonics.

loglevel *Command*

Syntax

lg | **loglevel [f ...]**

The types and amount of information logged with the system logging daemon can be configured. Without parameters, **loglevel** lists the possible log level settings and the current state of the connected server(s). See Section 3.6 on page 28 for a description of the logging options. Otherwise turn on the **f** log level setting(s). Appending a '**-**' turns off the level.

crashout *Command*

Syntax

c | **crashout** ***p t***

Set crashout time for product(s) *p* to *t* minutes. See **listp**.

timeout *Command*

Syntax

t | **timeout** ***p t***

Set the timeout for product(s) *p* to *t* minutes. See **listp**.

kill *Command*

Syntax

k | **kill** ***p n*** **[*n* ...]**

Revoke license(s) index *n* of product(s) *p*. The license will then be available to other users.

addnode *Command*

Syntax

n+ | **addnode** ***p host hostid*** [***host hostid ...***]

Add host machine(s) to the locked node list for product(s) *p*. *host* is the network host name for the required machine and *hostid* is its unique ID number. Note that the source of the unique ID depends on the make of the host machine.

You must ensure that the unique ID is given in the correct base. Note that there must be 1 or more node-locked licenses for this to have any effect. Node-locked licenses are available only on the listed hosts.

rmnode *Command*

Syntax

n- | **rmnode** ***p host*** [***host ...***]

Remove *host*(s) from locked node list of product(s) *p*. Only hosts previously added by **addnode** can be removed in this way. Note that all licenses for the product(s) will be revoked.

ftol *Command*

Syntax

f | **ftol** ***p n***

Convert *n* floating licenses to node-locked licenses for product(s) *p*. There must be at least *n* floating licenses available. Note that all licenses for the product(s) will be revoked.

ltof *Command*

Syntax

l | ltof *p n*

Convert *n* node-locked licenses to floating licenses for product(s) *p*. Only node-locked licenses previously converted with **ftol** can be moved in this way. All licenses for the product(s) will be revoked.

exhost *Command*

Syntax

e+ | exhost *p host* [*host* ...]

Exclude *host*(s) from running product(s) *p*. For example if a host is to be used exclusively as a server the system administrator may wish to prevent certain software from be used on it.

unexhost *Command*

Syntax

e- | unexhost *p host* [*host* ...]

Allow *host* to run product(s) *p*.

adduser *Command*

Syntax

ul+ | adduser *p user* [*user* ...]

Register *user*(s) to have a guaranteed license for product(s) *p*. This only has effect if the **USER_LOCKS** policy is in effect. See “Policies” on page 27.

remuser *Command*

Syntax

ul- | **remuser** ***p user*** **[*user ...*]**

Unregister *user*(s) from **USER_LOCKS** policy for product(s) *p*.

adduacc *Command*

Syntax

ua+ | **adduacc** ***p user lev*** **[*user lev ...*]**

Add *user* access levels. Set level to *lev* for user in product(s) *p*. Only effective if **ACCESS_LEVEL** policy set (see “Access levels” on page 28). Note: The *user* access level takes precedence over any group access level set.

remuacc *Command*

Syntax

ua- | **remuacc** ***p user*** **[*user ...*]**

Remove *user*(s) access level from **ACCESS_LEVEL** policy (that is, *user* will have full access or the group access level of their current group if any set).

addgacc *Command*

Syntax

ga+ | **addgacc** ***p group lev*** **[*group lev ...*]**

Add *group* access levels. Set level to *lev* for members of group in product(s) *p*. Only effective if **ACCESS_LEVEL** policy set. Note: The user access level takes precedence over any group access level set.

remgacc *Command*

Syntax

ga- | **remgacc** ***p group*** **[*group* ...]**

Remove *group*(s) access level from **ACCESS_LEVEL** policy (that is, members of the group will have full access unless modified by another group or user access level).

reinit *Command*

Syntax

ri | **reinit**

Reinitialize the connected server(s) and force it to re-read all permit files. All licenses for all products will be revoked. This can be used to add or change permits served by **hqn_lsd**.

3.3 Miscellaneous options

The following is a description of other miscellaneous commands or options available with **hqn_lsa**.

w *Command*

Syntax

w ***time***

Set the time (in seconds) that **hqn_lsa** will wait while trying to create a connection to a server or execute a command. The default time is ten seconds, minimum two seconds. Useful in initialization scripts, for example in the **rc.local** file.

y *Command*

Syntax

y

Suppress confirmations. Commands which normally ask for confirmation will continue as if '**yes**' was input.

lk *Command*

Syntax

lk ***lockstring***

Checks a permit file *lockstring* to ensure correct entry when changing a permit. Note: Place *lockstring* in quotes to protect special characters from the shell.

id *Command*

Syntax

id [*host* ...]

Returns the Internet address and unique ID number of the machine **hqn_lsa** is running on, and of the currently connected server. If host(s) are supplied **hqn_lsa** will try to obtain this information from an **hqn_lsd** server running on that host. For some makes of machine this is the only means of obtaining the unique ID number.

3.4 Policies

The policies flags for a product are held as a 32 bit value. The lower 16 bits are bit-flags for each of the policies described below. The upper 16 bits act as locks on each corresponding policy. If a lock bit is set, the corresponding lower bit cannot be changed by **hqn_lsa**. E.g. a value of **0x10001** sets the **GRAB** policy

and locks it. Using the 'policy' command allows the policies for a product to be configured either by a single value or a set of mnemonics. Attempts to change a locked policy will be ignored.

Current policies for the 'on' setting are described in Section 2.6 on page 13.

3.5 Access levels

If the ACCL policy is set, the system administrator can assign an access level to particular users and groups of users. Generally, full access is allowed unless the current user or group has a lower access level assigned. Access levels range from 0 to 8, where 8 allows full access, 1 allows minimum access, and 0 allows access to no software, and no license can be issued.

Note: The **ACCL** policy must be set otherwise all users will have full access.

3.6 Logging configuration

If required, all license transactions can be logged via the standard **syslog** daemon services. Information is logged as a **LOG_USER** facility at the **LOG_USER** or **LOG_WARNING** level. System administrators can tailor the types and amount of information logged with the **-loglevel** option. Log level switches control logged output as follows:

Bit	Switch	
1	**WARNING**	All warning and error messages — logged at system **LOG_WARN** level
2	**INFO**	Information messages, mainly during startup
3	**ADMIN**	Changes, via the administration tool **hqn_lsa**
4	**LICENSE**	All license transactions. Control output via the license bits 5-10 (listed below)

Table 3.1

Bit	Switch	
5	**PRODUCT**	Output the full product name on each transaction.
6	**HOSTNAME**	Output the name of the product host
7	**HOSTADDR**	Output the Internet address of the product host
8	**USERNAME**	Output the name of the user of the product
9	**UID**	Output the user ID
10	**EXTERR**	Output extra information on some license 'error' conditions, for example, redirected requests. May be best off on backup servers
11	**DOMAIN**	Messages relating to domain initialization and validation. Generally only required for checking unusual conditions

Table 3.1

Log settings are turned off by appending a '-' to the switch name. For example,

```
hqn_lsa -lg exterr-
```

turns off extra error logging. In addition some standard settings are provided, as listed below. Each standard setting has a configuration of log settings automatically switched on.

Name	Log settings on
DEFAULT	**WARNING INFO ADMIN LICENSE PRODUCT HOSTNAME USERNAME**
SMALL	**WARNING INFO ADMIN LICENSE HOSTNAME USERNAME**

Table 3.2

Name	Log settings on
FULL	**DEFAULT HOSTADDR UID EXTERR**
MINIMAL	**WARNING ADMIN LICENSE HOSTADDR UID**
EXTRA	**FULL DOMAIN**
NONE	No logging

Table 3.2

hqn_lsd starts up with the default standard setting, plus extra error logging turned on. The **-f** option on **hqn_lsd**'s command line may be used to turn on domain logging during startup.