LispWorks®

Release Notes and Installation Guide

Version 5.0

Copyright and Trademarks

LispWorks Release Notes and Installation Guide

Version 5.0

July 2006

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: 305 468 5262

From elsewhere: +44 870 2206189

www.lispworks.com

Contents

1

Introduction

1.1 LispWorks Editions

LispWorks is available in four product editions on the Mac OS X, Windows, Linux and FreeBSD platforms:

- Personal Edition - includes a native graphical IDE and COM / Automation on Windows.
- Professional Edition – includes all Personal Edition features, plus Application Delivery, CLIM 2.0 on X11 / Motif and Windows, and a X11 / Motif GUI on Mac OS X.
- Enterprise Edition – includes all Professional Edition features, plus KnowledgeWorks, LispWorks ORB, and Common SQL.
- Academic Edition - includes all Enterprise Edition features except Application Delivery.

Note: on Solaris and HP-UX LispWorks is licensed differently to other platforms, as detailed in “LispWorks for UNIX” on page 3.

1.1.1 Personal Edition

The LispWorks Personal Edition, distributed free of charge, allows you to explore a fully enabled Common Lisp programming environment and to develop small- to medium-scale programs for personal and academic use.

Please note, however, that the Personal Edition has the following limitations designed to prevent commercial exploitation of this free product:

- There is a heap size limit which, if exceeded, causes the image to exit. A warning is provided when the limit is approached. On Mac OS X, at this point there is an option to abort all processes, allowing you to stop consing and save your work before you actually reach the heap size limit.

  Note: the higher heap limit introduced in version 4.4.6 has been retained in version 5.0.

- There is a time limit of 5 hours for each session, after which the image is exited. You are warned after 4 hours of use.

  Note: If the image is dismissed because of heap size or time limitations, there are no exit cleanups. For example, you are not asked about saving changed buffers. Also, working and temporary files are not removed. In particular, this may result in `.htm` files generated by manual searches being left behind in `/tmp` or your `TEMP` folder.

- The functions `save-image`, `deliver`, and `load-all-patches` are not available.

- Initialization files are not available.

- Professional and Enterprise Edition module loading is not included in the Personal Edition.

1.1.2 Professional Edition

LispWorks 5.0 Professional Edition includes the following features:

- Fully supported commercial product
- Delivery of commercial end-user applications
- CLIM 2.0 on X11 / Motif and Windows

- 30-day free “Getting Started” technical support

1.1.3 Enterprise Edition

LispWorks 5.0 Enterprise Edition provides further support for the software needs of the modern enterprise, including:

- All the features of the Professional Edition
- Database access through the Common SQL interface
- Portable distributed computing through CORBA
- Expert systems programming through KnowledgeWorks and embedded Prolog compiler

1.1.4 Academic Edition

LispWorks 5.0 Academic Edition offers a solution for teaching and research institutions which require multiple seats at an economic price. It includes:

- All the features of the Enterprise Edition except delivery of commercial end-user applications
- Unlimited number of users at one site

1.2 LispWorks for UNIX

On Solaris and HP-UX the Edition model described above does not apply.

LispWorks for UNIX 5.0 is available with a basic developer license, and the add-on products CLIM, KnowledgeWorks, LispWorks ORB and Application Delivery are each separately available.

1.3 Further details

For further information about LispWorks products visit

`www.lispworks.com`

To purchase LispWorks please follow the instructions at:

`www.lispworks.com/buy`

1.4 About this Guide

This document is an installation guide and release notes for LispWorks 5.0 on Mac OS X, Windows, Linux, FreeBSD and UNIX platforms. It also explains how to configure LispWorks to best suit your local conditions and needs.

This guide provides instructions for installing and loading the modules included with each Edition or add-on product.

1.4.1 Installation and Configuration

The next four chapters explain in brief and sufficient terms how to complete a LispWorks installation on Mac OS X, Windows, Linux or UNIX. Choose the chapter for your platform: Chapter 2, “Installation on Mac OS X”, Chapter 3, “Installation on Windows” or Chapter 4, “Installation on Linux” or Chapter 6, “Installation on UNIX”.

The following four chapters explain in detail everything necessary to configure, run, and test LispWorks 5.0. Choose the chapter for your platform: Chapter 7, “Configuration Details on Mac OS X”, Chapter 8, “Configuration Details on Windows”, Chapter 9, “Configuration Details on Linux and FreeBSD” or Chapter 10, “Configuration Details on UNIX”. This also includes sections on initializing LispWorks and loading some of the modules. You should have no difficulty configuring, running, and testing LispWorks using these instructions if you have a basic familiarity with your operating system and Common Lisp.

1.4.2 Troubleshooting

Chapter 11, “Troubleshooting, Patches and Reporting Bugs”, discusses other issues that may arise when installing and configuring LispWorks. It includes a section that provides answers to problems you may have encountered, sections on the LispWorks patching system (used to allow bug fixes and private patch changes between releases of LispWorks), and details of how to report any bugs you encounter.

1.4.3 Release Notes

Chapter 12, “Release Notes”, highlights what is new in this release and special issues for the user’s consideration.

2

Installation on Mac OS X

This chapter is an installation guide for LispWorks 5.0 for Macintosh. Chapter 7 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

2.1 Choosing the Graphical User Interface

LispWorks for Macintosh supports two different graphical interfaces. Most users choose the native Mac OS X GUI, but you can use the Motif GUI instead.

Different executables and supporting files are supplied for the two options. You need to decide at installation time which of the GUIs you will be using, or decide to install support for both. If you install just one GUI option and later decide to install the other GUI option, you can simply run the installer again.

LispWorks for Macintosh Personal Edition supports only the native Mac OS X GUI.

64-bit LispWorks for Macintosh only supports the Motif GUI.

2.2 Documentation

The LispWorks documentation set is included in two electronic forms: HTML and PDF. You can chose whether to install it as described in Section 2.4, “Installing LispWorks for Macintosh”.

The HTML version can be used from within the LispWorks environment via the **Help** menu. You will need a suitable web browser installed. You can also reach the HTML documentation at the page **`manual/online/intro.htm`** in the LispWorks library. If you choose not to install the documentation, you will not be able to access the Browsable (HTML) Documentation from the LispWorks **Help** menu.

The PDF version is suitable for printing. Each manual in the documentation set is presented in a separate PDF file in the LispWorks library under **`manual/offline/pdf`**. To view and print these files, you will need a PDF viewer such as Adobe® Reader®. This can be downloaded from the Adobe website at **`www.adobe.com`**.

2.3 Software and hardware requirements

LispWorks 5.0 is a universal binary, which supports Macintosh computers containing either PowerPC or Intel CPUs.

An overview of system requirements is provided in Table 2.1. The sections that follow discuss any relevant details.

Hardware Requirements	Software Requirements
(32-bit) 32MB of memory, preferably 64MB	Mac OS X version 10.3.x or 10.4.x.
(64-bit) G5 processor, 64MB of memory, preferably 128MB	Mac OS X 10.4.5 or higher.
200MB of disk space for 32-bit Enterprise Edition plus documentation	OpenMotif 2.3 if you want to run the X11/Motif GUI.

Table 2.1

Hardware Requirements	Software Requirements
140MB of disk space for 64-bit Enterprise Edition plus documentation	

Table 2.1

2.4 Installing LispWorks for Macintosh

2.4.1 Main installation and patches

LispWorks Professional, Enterprise and Academic Editions are supplied as an installer containing version 5.0. There may be a downloadable patch bundle which upgrades LispWorks to version 5.0.x. You need to complete the main installation before adding patches.

LispWorks Personal Edition is supplied as an installer containing version 5.0.

2.4.2 Information for Beta testers

Users of LispWorks 5.0 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 5.0. You can run the Beta installer and select the **Uninstall** option (and then remove any patches) or simply drag the **`LispWorks 5.0`** folder to the trash.

2.4.3 Information for LispWorks 4.4 users

You can install LispWorks 5.0 in the same location as LispWorks 4.4.5 or the Xanalys LispWorks 4.4 release. If you always choose the default install location, a new **`LispWorks 5.0`** folder will be created alongside the other versions.

Similarly LispWorks Personal Edition 5.0 can be installed in the same location as previous versions.

2.4.4 Use an adminstrator account

To install LispWorks in the default installation location under **`/Applications`** you must log on as an administrator.

However, a non-administrator may install LispWorks elsewhere.

2.4.5 Launch the LispWorks installer

If you have downloaded LispWorks, you may need to mount the disk image containing the installer. This is called **LispWorks-5.0.dmg** or **LispWorks64bit-5.0.dmg** — simply double-click on the **.dmg** file to mount it.

If you have received LispWorks on a CD-ROM, insert the disk in a drive and double-click on the disk icon to mount it.

To install LispWorks (32-bit) for Macintosh, open the **macos** folder and double-click on the **LispWorks_Installer** application to launch it.

To install LispWorks (64-bit) for Macintosh, open the **darwin64** folder and double-click on the **LispWorks64bit_Installer** application to launch it.

Note: the names of the installer and downloadable file will vary slightly for the Personal and Academic Editions.

2.4.6 The Read Me

The Read Me presented next by the installer is a plain text version of this *LispWorks Release Notes and Installation Guide*.

2.4.7 The License Agreement

Check the license agreement. You need to actually read this to the end, then click **Continue**. You will be asked if you agree to the license terms. Click the **Accept** button only if you accept the terms of the license. If you click **Disagree**, then the installer will not proceed.

2.4.8 Select Destination

All the files installed with LispWorks are placed in the LispWorks folder, which is named **LispWorks 5.0**, **LispWorks 5.0 (64-bit)**, **LispWorks Personal 5.0** or **LispWorks Academic 5.0** depending on which edition you are installing. By default, the LispWorks folder is placed in the main **Applications** folder but you can choose an alternative location during installation by clicking the **Select Folder...** button.

Click **Continue** after selecting a folder.

Note: The **Applications** folder may display in the Finder with a name localized for your language version of Mac OS X.

2.4.9 Choose your installation type

2.4.9.1 The native Mac OS X GUI (only on 32-bit LispWorks)

If you simply want to install LispWorks for the native Mac OS X GUI with Aqua look and feel, and to install the documentation, choose **Easy Install**.

Note: Choosing **Easy Install** in the 64-bit LispWorks installer will install the X11 / Motif GUI.

2.4.9.2 The X11/Motif GUI (required on 64-bit LispWorks)

If you want to install LispWorks with the X11 / Motif GUI, choose **Custom Install** and select the option "LispWorks with X11 / Motif IDE".

Note: to run LispWorks with the X11 / Motif GUI, you will need both of these installed:

- An X server such as Apple's X11.app, available at **www.apple.com**.
- OpenMotif 2.3. For availability see "Obtaining OpenMotif" on page 13.

Neither X11 or Motif are required at the time you install LispWorks, however.

The X11 / Motif GUI is not available for the Personal Edition.

2.4.9.3 The Documentation

If you use **Easy Install** the documentation will be installed.

If you do not wish to install the documentation, use **Custom Install** and uncheck the "LispWorks Documentation" option.

For LispWorks Personal Edition, the documentation is supplied as an optional extra download and is not part of the main installer. You should install it in the same location as the LispWorks Personal Edition software: then the soft-

ware and the documentation will reside in the same **LispWorks Personal 5.0** folder.

2.4.10 Installing

Now click **Install**.

2.4.11 Enter License Data

Enter your serial number and license key when the installer asks for these details.

If you have received a media kit then your LispWorks license key is supplied on a label on the folder containing the CD-ROM. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

Contact **lisp-keys@lispworks.com** if you have problems with your LispWorks license key.

2.4.12 Add LispWorks to the Dock (only on 32-bit LispWorks)

If you are installing the native Mac OS X LispWorks GUI, the installer asks if you wish to add LispWorks to the Mac OS X Dock. Click **OK** if you anticipate launching LispWorks frequently, or choose not to add LispWorks to the Dock by clicking **Cancel**.

Note: On Mac OS X 10.4, LispWorks may not be visible in the Dock until you restart the computer or log out and then log back in.

2.4.13 Finishing up

You should now see a message confirming that installation of LispWorks was successful. Click the **Quit** button.

Note: LispWorks needs to be able find its library at runtime and therefore the LispWorks installation should not be moved around piecemeal. If you must move it, move the entire LispWorks installation folder. If you simply want to run LispWorks from somewhere more convenient, then consider adding a shortcut.

2.4.14 Installing Patches

After completing the main installation of the Professional or Enterprise Edition, ensure you install the latest patches which are available for download at **www.lispworks.com/downloads/patch-selection.html**. Patch installation instructions are in the README file accompanying the patch download.

2.4.15 Obtaining OpenMotif

LispWorks for Macintosh 5.0 on X11/Motif requires Open Motif 2.3.

The library for 32-bit LispWorks is **/usr/local/lib/libXm.4.dylib**. You can can build a PowerPC or Intel binary from the sources at **www.motifzone.net**.

Apple do not currently support a 64-bit X11 installation. Contact Lisp Support if you need a library suitable for 64-bit LispWorks.

2.5 Starting LispWorks for Macintosh

2.5.1 Start the native Mac OS X LispWorks GUI

Assuming you have installed this option, you can now start LispWorks with the native Mac OS X GUI by double-clicking on the LispWorks icon in the LispWorks folder.

Note: The LispWorks folder is described in “Select Destination” on page 10.

If you added LispWorks to the Dock during installation, you can also start LispWorks from the Dock. If you did not add LispWorks to the Dock during installation, you can add it simply by dragging the LispWorks icon from the Finder to the Dock.

If you want to create a LispWorks image which does not start the GUI automatically, you should use a configuration script that calls

```
(save-image ... :environment nil)
```

and pass it to the supplied **lispworks-5-0-0-macos-universal** image.

See Section 7.3, “Configuring your LispWorks installation” for more information about configuring your LispWorks image for your own needs.

Note: for the Personal Edition, the folder name and icon name are LispWorks Personal, the image is **lispworks-personal-5-0-0-macos-universal**, and **save-image** is not available.

Note: for the Academic Edition, the folder name and icon name are Academic Edition, and the image is called **lispworks-academic-5-0-0-macos-universal**.

2.5.2 Start the X11/Motif LispWorks GUI

Assuming you have installed this option, and that you have X11 running and Motif installed, you can now start LispWorks with the X11 / Motif GUI.

Note that the supplied image does not start its GUI automatically by default. There are three alternate ways to make the GUI start:

1. Call the function **env:start-environment**

   Follow this session in the X11 terminal (xterm by default):

   ```
   xterm% cd "/Applications/LispWorks 5.0"
   xterm% ./lispworks-5-0-0-macos-universal-motif
   LispWorks(R): The Common Lisp Programming Environment
   Copyright (C) 1987-2006 LispWorks Ltd.  All rights reserved.
   Version 5.0.0
   Saved by LispWorks as lispworks-5-0-0-darwin-motif, at 21 Jul
   2006 12:57
   User dubya on octane
   ; Loading text file /Applications/LispWorks 5.0/Library/lib/5-0-
   0-0/config/siteinit.lisp
   ;  Loading text file /Applications/LispWorks 5.0/Library/lib/5-0-
   0-0/private-patches/load.lisp
   ; Loading text file /u/ldisk/dubya/.lispworks

   CL-USER 1 > (env:start-environment)
   ```

   The LispWorks X11 / Motif IDE and Lisp Monitor window should appear.

   You may put the call to **env:start-environment** at the end of your initialization file, if desired.

2. Pass the **-env** command line argument

   The **-env** command line argument causes the function **env:start-environment** to be called.

Follow this session in the X11 terminal:

```
xterm% cd "/Applications/LispWorks 5.0"
xterm% ./lispworks-5-0-0-macos-universal-motif -env
```

The LispWorks X11/Motif IDE and Lisp Monitor window should appear.

3. Create an image which starts the GUI automatically

   If you want to create a LispWorks image which starts the GUI automatically, you should make a configuration script that calls

   `(save-image ... :environment t)`

   and pass it to the supplied **`lispworks-5-0-0-macos-universal-motif`** image. Note: This will create a non-universal binary, containing only the architecture on which you call **`save-image`**.

   See Section 7.3, “Configuring your LispWorks installation” for more information about configuring your LispWorks image for your own needs.

Note: the Academic Edition X11/Motif image is called **`lispworks-academic-5-0-0-macos-universal-motif`**.

3

Installation on Windows

This chapter is an installation guide for LispWorks 5.0 for Windows and LispWorks 5.0 (64-bit) for Windows. Chapter 8 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

3.1 Documentation

The LispWorks documentation set is available in two electronic forms: HTML and PDF. You can choose whether to install either of these.

If you install the HTML documentation, then it can be used from within the Common LispWorks environment via the **Help** menu. It is also available from the **Start** menu under **Start > All Programs > LispWorks 5.0 > HTML Documentation**.

The PDF version is suitable for printing. Each manual in the documentation set is presented in a separate PDF file, available from the **Start** menu under **Start > All Programs > LispWorks 5.0 > PDF Documentation**. To view and print these files, you will need a PDF viewer such as Adobe® Reader®. If you do not already have this, it can be downloaded from the Adobe website.

3.2 Installing LispWorks for Windows

3.2.1 Main installation and patches

LispWorks Professional, Enterprise and Academic Editions are supplied as an installer containing version 5.0. There may be a downloadable patch bundle which upgrades LispWorks to version 5.0.x. You need to complete the main installation before adding patches.

LispWorks Personal Edition is supplied as an installer containing version 5.0.

3.2.2 Visual Studio runtime components and Windows Installer

On systems where this is not present, installing LispWorks will automatically install a copy of the Microsoft.VC80.CRT component, which contains the Microsoft Visual Studio runtime DLLs needed by LispWorks.

It will also automatically install Windows Installer 3.1 when needed (for example on Windows 2000).

3.2.3 Installing over previous versions

You can install LispWorks 5.0 in the same location as LispWorks 4.4.5. This is the default installation location.

You can also install LispWorks 5.0 without uninstalling older versions such as Xanalys LispWorks 4.4 or Xanalys LispWorks 4.3 provided that the chosen installation directory is different.

The LispWorks Personal Edition installation behaves in the same way. Ensure that, if you choose this option, you install the LispWorks Personal Edition 5.0 documentation in the same location as the software.

3.2.4 Information for Beta testers

Users of LispWorks 5.0 Beta should completely uninstall it before installing LispWorks 5.0. Remember to remove any patches added since the initial beta release.

3.2.5 To install the Professional or Enterprise Edition

To install LispWorks (32-bit) for Windows run **x86-win32\Setup.exe**.

To install LispWorks (64-bit) for Windows run **x64-windows\Setup.exe**.

Follow the instructions on screen and read the remainder of this section.

3.2.5.1 Entering the License Data

Enter your serial number and license key when the installer asks for these details.

If you have received a media kit then your LispWorks license key is supplied on a label on the folder containing the CD-ROM. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

Contact **lisp-keys@lispworks.com** if you have problems with your LispWorks license key.

3.2.5.2 Installing the Documentation

During the installation, you will need to decide whether to install the Browsable Documentation. If you choose not to install this, you will be able to access the Browsable (HTML) Documentation only from **www.lispworks.com**.

You will also need to decide whether to install the Printable Documentation. Individual PDF files can also be downloaded from **www.lispworks.com.**

3.2.5.3 Installing Patches

After completing the main installation of the Professional or Enterprise Edition, ensure you install the latest patches which are available for download at **www.lispworks.com/downloads/patch-selection.html**. Patch installation instructions are in the README file accompanying the patch download.

3.2.5.4 Starting LispWorks

When the installation is complete, you can start LispWorks by choosing **Start > All Programs > LispWorks 5.0 > LispWorks**.

Note: LispWorks needs to be able find its library at runtime and therefore the LispWorks installation should not be moved around piecemeal. If you simply want to run LispWorks from somewhere more convenient, then consider adding a shortcut.

3.2.6 To install the Academic Edition

Installation is just as described in Section 3.2.5, except that the default installation location is **`C:\Program Files\LispWorks 5.0 Academic`**.

Note: do not attempt to to install different editions in the same location, since some filenames coincide and uninstallation may break.

When the installation is complete, you can start LispWorks by choosing **Start > All Programs > LispWorks 5.0 Academic > LispWorks**.

4

Installation on Linux

This chapter is an installation guide for LispWorks 5.0 for Linux and LispWorks 5.0 (64-bit) for Linux. Chapter 9, discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

4.1 Software and hardware requirements

An overview of system requirements is provided in Table 4.1. The sections that follow discuss any relevant details.

Hardware Requirements	Software Requirements
32MB of memory, preferably 64MB	RedHat Linux (version 9 or later) or a distribution with kernel version 2.4 or later that supports NPTL and glibc 2.3.2 or later.

Table 4.1

Hardware Requirements	Software Requirements
118MB of disk space for Enterprise Edition (32-bit) plus documentation	OpenMotif 2.2 (or higher) or Lesstif 0.93.18 (or higher)
158MB of disk space for Enterprise Edition (64-bit) plus documentation	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 4.1

4.1.1 Motif libraries

LispWorks 5.0 for Linux requires that the X11 release 6 (or higher) and either OpenMotif (version 2.2 or higher) or Lesstif 0.93.18 (or higher, 32-bit only) are installed on your machine. If you need to install the graphical library first, contact your systems administrator for help.

OpenMotif is available from **`www.motifzone.net`** and Lesstif is available from **`www.lesstif.org`**.

Note: In order for the LispWorks IDE to run "out of the box", Lesstif or Motif must be installed on the target machine.

Note: You should be able to run LispWorks 5.0 and LispWorks 4.4 simultaneously with either OpenMotif or Lesstif installed.

4.1.2 Disk requirements during installation

LispWorks requires about 46MB for 32-bit and 58MB for 64-bit to install without documentation. Installing the documentation adds about 94MB (Professional / Enterprise / Academic) or 45MB (Personal) to this. A full installation of the Enterprise Edition with all documentation and loadable modules requires about 154MB.

The Professional / Enterprise / Academic documentation includes printable PDF format manuals. You may delete any of these that you do not need. They

are available at **www.lispworks.com/documentation** in any case, and the same manuals are also available there in PostScript format.

4.1.3 Memory requirements at runtime

LispWorks 5.0 requires an absolute minimum of 32MB of user RAM in order to run. More RAM minimizes swapping and therefore improves performance. With 64MB RAM, LispWorks will perform well.

For information about the behavior of LispWorks near the limit of real memory, see “Memory requirements” on page 94.

4.2 License agreement

Before installing, you must read and agree to the license terms. To do this, mount the CD-ROM on your CD-ROM drive and **cd** to the directory containing the product you wish to install.

For LispWorks (32-bit) for Linux the directory is **x86-linux**.

For LispWorks (64-bit) for Linux the directory is **amd64-linux**.

Now run the one of following scripts.

Note: You must run this script as the same user that later performs the installation. In particular, if you are going to install LispWorks from the RPM file, you must run the license script while logged on as root.

- For the Personal Edition, run:

  ```
  sh lwlper-license.sh
  ```

  and enter “yes” if you agree to the license terms.

- For the Professional and Enterprise Editions, run

  ```
  sh lwl-license.sh
  ```

  and enter “yes” if you agree to the license terms.

- For the Academic Edition, run

  ```
  sh lwlac-license.sh
  ```

  and enter “yes” if you agree to the license terms.

4.3 Software on the CD-ROM

LispWorks 5.0 for Linux is supplied on a CD-ROM in two different formats: RedHat Package Management (RPM) files and **tar** files. RPM is a utility like **tar**, except it can actually install products after unpacking them. See Section 4.4.3 for more information. Both formats are in the **x86-linux** and **amd64-linux** directories on your CD-ROM.

4.3.1 Professional and Enterprise Edition distributions

The CD-ROM contains all of the relevant modules for your Professional or Enterprise Edition. The separately installable modules installed with LispWorks are: CLIM 2.0, KnowledgeWorks, LispWorks ORB, and Common SQL. Section 1.1 provides Edition details.

The package name for the Professional/Enterprise Edition is **lispworks** and the separately installable modules are:

```
lispworks-clim
lispworks-kw
lispworks-corba
lispworks-sql
```

The installation instructions provide the names of the individual distribution files.

4.3.2 Academic Edition distributions

The CD-ROM contains all of the relevant modules for your Academic Edition. The separately installable modules installed with LispWorks are: CLIM 2.0, KnowledgeWorks, LispWorks ORB, and Common SQL. Section 1.1 provides Edition details.

The package name for the Academic Edition is **lispworks-academic** and the separately installable modules are:

```
lispworks-academic-clim
lispworks-academic-kw
lispworks-academic-corba
lispworks-academic-sql
```

The installation instructions provide the names of the individual distribution files.

4.3.3 Personal Edition distribution

You can install the LispWorks Personal Edition by downloading it from the LispWorks Web site at **www.lispworks.com/downloads**.

The package name for the Personal Edition is **lispworks-personal**.

4.4 Installing LispWorks for Linux

4.4.1 Main installation and patches

LispWorks Professional, Enterprise and Academic Editions are supplied as an installer containing version 5.0. There may be a downloadable patch bundle which upgrades LispWorks to version 5.0.x. You need to complete the main installation before adding patches.

LispWorks Personal Edition is supplied as an installer containing version 5.0.

4.4.2 Information for Beta testers

Users of LispWorks 5.0 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 5.0.

See “Uninstalling LispWorks for Linux” on page 33 for instructions.

4.4.3 Installation from the binary RPM file

We recommend that you use RPM 4.3 or later (however see below for problems with **--prefix** argument with some versions of RPM). The distribution files are also provided in **tar** format in case you do not have a suitable version of RPM or are using another distribution of Linux.

If you already have LispWorks 5.0 Beta installed, please uninstall it before installing this product. See Section 4.9, “Uninstalling LispWorks for Linux”.

Some versions of RPM may cause problems (eg. RPM 3.0). If you get the following message when using the **--prefix** argument:

```
rpm: only one of --prefix or --relocate may be used
```

try upgrading to RPM 3.0.2 or greater.

Installation of LispWorks for Linux from the RPM file must be done while you are logged on as root.

4.4.3.1 Installation directories

By default 32-bit LispWorks is installed in **/usr/lib/LispWorks** and a symbolic link to the executable is placed in **/usr/bin/lispworks-5-0-0-x86-linux**. Similarly, 64-bit LispWorks is installed in **/usr/lib64/LispWorks** and a symbolic link to the executable is placed in **/usr/bin/lispworks-5-0-0-amd64-linux**. However, the RPM is relocatable, and the **--prefix** option can be used to allow the installation of LispWorks in a user-specified directory. The default prefix is **/usr**.

Note: RPM version 4.2 has bug which can hinder secondary installations (CLIM, Common SQL, LispWorks ORB or KnowledgeWorks) in a in a user-specified directory. See “RPM_INSTALL_PREFIX not set” on page 97 for a workaround.

Note: the Personal and Academic Editions by default install in **/usr/lib/LispWorksPersonal** and **/usr/lib/LispWorksAcademic** respectively. Do not attempt to to install different editions in the same location, since some filenames coincide and uninstallation may break.

4.4.3.2 Selecting the correct RPM files

The main RPM file in the LispWorks distribution is named using the following pattern

lispworks-5.0-*n*.*arch*.rpm

The integer *n* denotes a build number and will be same in all files in your distribution. The string *arch* will be either **i386** for 32-bit LispWorks or **x86_64** for 64-bit LispWorks. The text below assumes 32-bit LispWorks.

Note: For the Personal Edition, use **lispworks-personal-5.0-*.i386.rpm** wherever **lispworks-5.0-*.i386.rpm** is mentioned in this document. See Section 1.1.1, “Personal Edition” for more information specific to the Personal

Edition. Similarly, for the Academic Edition use **lispworks-academic-5.0-*.i386.rpm**.

4.4.3.3 Installing or upgrading LispWorks for Linux

To install or upgrade LispWorks from the RPM file, perform the following steps as root:

1. Locate the RPM installation file **lispworks-5.0-*n*.i386.rpm**.
2. Install or upgrade LispWorks in the standard RPM way, for example:

   rpm --install lispworks-5.0-*n*.i386.rpm

   This command installs LispWorks in **/usr/lib/LispWorks**. A command line of the form

   rpm --install --prefix <*directory*> lispworks-5.0-*n*.i386.rpm

   installs LispWorks in <*directory*>.

The directory name must be an absolute pathname. Relative pathnames and pathnames including shell-expanded characters such as **.** and **~** do not work.

Note: LispWorks needs to be able find its library at runtime and therefore the LispWorks installation should not be moved around piecemeal. If you simply want to run LispWorks from somewhere more convenient, then consider adding a symbolic link.

See Section 4.6 for instructions on entering your license details.

4.4.3.4 Installing the loadable Professional and Enterprise Edition modules

The following modules are packaged as separate RPM files for installation after the main **lispworks** package.

File Distribution	Layered Product	LispWorks Edition
lispworks-clim-5.0-*n*.i386.rpm	CLIM 2.0	Professional

Table 4.2 File distributions for layered products in the Professional and Enterprise Editions

File Distribution	Layered Product	LispWorks Edition
lispworks-kw-5.0-*n*.i386.rpm	KnowledgeWorks	Enterprise
lispworks-corba-5.0-*n*.i386.rpm	LispWorks ORB	Enterprise
lispworks-sql-5.0-*n*.i386.rpm	Common SQL	Enterprise

Table 4.2 File distributions for layered products in the Professional and Enterprise Editions

Install these modules by substituting the above file names into the same commands you used to install the LispWorks package (Section 4.4.3.3).

If you used a **--prefix** argument when installing LispWorks, then use the same prefix for these modules.

4.4.3.5 Installing the loadable Academic Edition modules

The following modules are packaged as separate RPM files for installation after the main **lispworks-academic** package:.

File Distribution	Layered Product	LispWorks Edition
lispworks-academic-clim-5.0-*n*.i386.rpm	CLIM 2.0	Academic
lispworks-academic-kw-5.0-*n*.i386.rpm	KnowledgeWorks	Academic
lispworks-academic-corba-5.0-*n*.i386.rpm	LispWorks ORB	Academic
lispworks-academic-sql-5.0-*n*.i386.rpm	Common SQL	Academic

Table 4.3 File distributions for layered products in the Academic Edition

Install these modules as described in Section 4.4.3.4.

4.4.3.6 Documentation and saving space

Documentation in HTML format is provided with all editions. Additionally, PDF format is provided as part of the Professional/Enterprise and Academc distributions, and PostScript format is available to download. To obtain copies of the printable manuals, see Section 4.8, "Printable LispWorks documentation".

Documentation is installed by default in the **lib/5-0-0-0/manual** sub-directory of the LispWorks installation directory.

Using RPM, you can save space by choosing not to install the documentation. For example, use the following command (all on one line):

```
rpm --install --excludedocs --prefix <directory>
lispworks-5.0-n.i386.rpm
```

To install the documentation at a later stage, you need to use the **--replacepkgs** option:

```
rpm --install --prefix <directory> --replacepkgs
lispworks-5.0-n.i386.rpm
```

4.4.3.7 Installing Patches

After completing the main RPM installation of the Professional or Enterprise Edition and any modules, ensure you install the latest patches from the RPM file available for download at **www.lispworks.com/downloads/patch-selection.html**. Patch installation instructions are in the README file accompanying the patch download.

4.4.4 Installation from the tar files

The LispWorks distribution is also provided as **tar** files compressed using **gzip** for use if you do not have an appropriate version of RPM to unpack the RPM binary file. The gzipped files for the various Editions are as follows:

Table 4.4 Files for Professional and Enterprise

lw50-x86-linux.tar.gz	32-bit LispWorks Professional and Enterprise image, modules and examples
lw50-amd64-linux.tar.gz	64-bit LispWorks Professional and Enterprise image, modules and examples
lwdoc50-x86-linux.tar.gz	Documentation in HTML and PDF formats

Table 4.5 Files for the Personal Edition

lwper50-x86-linux.tar.gz	LispWorks Personal image, modules and examples
lwperdoc50-x86-linux.tar.gz	Documentation in HTML format

Table 4.6 Files for the Academic Edition

lwac50-x86-linux.tar.gz	LispWorks Academic image, modules and examples
lwdoc50-x86-linux.tar.gz	Documentation in HTML and PDF formats

To install from these files:

1. Follow the instructions under Section 4.2, “License agreement”.
2. Use **cd** to change directory to the location of the tar files before running the installation script.
3. Run the installation script **lwl-install.sh** (or **lwlper-install.sh** for the Personal Edition, **lwlac-install.sh** for the Academic Edition)).

This script takes **--prefix** and **--excludedocs** arguments like **rpm** to control the installation directory and amount of documentation installed.

For example, to install the 32-bit Professional Edition in **/usr/lisp-works**, without documentation, from a CD-ROM mounted on **/mnt/cdrom1** you would use:

```
cd /mnt/cdrom1/x86-linux
sh lwl-install.sh --excludedocs --prefix /usr/lispworks
```

Note: the default location under **/usr/local** is appropriate for this unmanaged (non-RPM) installation.

See Section 4.6 for how to enter your license details.

4.4.4.1 Installing Patches

After completing the main **tar** installation of the Professional or Enterprise Edition, ensure you install the latest patches from the **tar** archive available for download at **www.lispworks.com/downloads/patch-selection.html**. Patch installation instructions are in the README file accompanying the patch download.

4.5 LispWorks looks for a license key

If you installed the Professional, Enterprise or Academic Edition of LispWorks, the image looks for a valid license key. If you try to run these LispWorks Editions without a valid key, a message prints reporting that no valid key was found.

For instructions on entering your license key, see Section 4.6.1, “Entering the license data” below.

For more information about license keys, see Section 9.2, “License keys”.

4.6 Running LispWorks

The LispWorks executable is located in **/usr/lib/LispWorks** or **/usr/lib64/LispWorks** directory of the installation (assuming the default prefix of **/usr**) and should not be moved without being resaved because it needs

to be able to locate the corresponding library directory on startup. There is also a symbolic link from the **/usr/bin** directory.

The LispWorks executable is named as shown here:.

lispworks-personal-5-0-0-x86-linux	Personal Edition
lispworks-5-0-0-x86-linux	32-bit Professional or Enterprise Edition
lispworks-5-0-0-amd64-linux	64-bit Enterprise Edition
lispworks-academic-5-0-0-x86-linux	Academic Edition

When you run LispWorks, the Lisp Monitor and splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 93 for details if this does not happen.

4.6.1 Entering the license data

When you run the LispWorks Professional / Enterprise / Academic Edition for the first time, you will need to enter your license details. This should be done as follows (all on one line):

lispworks-5-0-0-x86-linux --lwlicenseserial *SERIALNUMBER* --lwlicensekey *LICENSEKEY*

where *SERIALNUMBER* and *LICENSEKEY* are the strings supplied with LispWorks.

If you have received a media kit then your LispWorks license key is supplied on a label on the folder containing the CD-ROM. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

Contact **lisp-keys@lispworks.com** if you have problems with your LispWorks license key.

4.7 Configuring the image

If you installed the Professional, Enterprise or Academic Edition of LispWorks, you can now configure your LispWorks image to suit your needs

and load the Professional or Enterprise Edition modules as necessary. For instructions, see Chapter 9, “Configuration Details on Linux and FreeBSD”.

Details about the Personal Edition can be found in “Personal Edition” on page 2.

4.8 Printable LispWorks documentation

In a default Professional / Enterprise / Academic installation, the **lib/5-0-0-0/manual/offline** directory contains PDF format versions of the manuals.

In the Personal Edition, these files are omitted to reduce installer download time, but may be freely downloaded if required from **www.lispworks.com/documentation**.

PostScript format versions of the manuals are also available for download.

4.9 Uninstalling LispWorks for Linux

A RPM installation of LispWorks can be uninstalled in the usual way, for example by executing:

```
rpm --erase lispworks-5.0
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named **lispworks-patches5.0**. The same applies to additional RPM packages such as **lispworks-corba**.

If patches have been added from a tar archive, you will need to remove them by hand.

If you installed LispWorks from the tar archives, simply do

```
rm -rf /usr/local/lib/LispWorks
```

5

Installation on FreeBSD

This chapter is an installation guide for LispWorks 5.0 for FreeBSD. Chapter 9, discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

5.1 Software and hardware requirements

An overview of system requirements is provided in Table 5.1. The sections that follow discuss any relevant details.

Hardware Requirements	Software Requirements
32MB of memory, preferably 64MB	FreeBSD 5.4 or later, or FreeBSD 6.0 or later with compat5x
111MB of disk space for Enterprise Edition plus documentation	OpenMotif 2.2 (or higher)

Table 5.1

Hardware Requirements	Software Requirements
	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 5.1

5.1.1 Motif libraries

LispWorks 5.0 for FreeBSD requires that the X11 release 6 (or higher) OpenMotif (version 2.2 or higher) are installed on your machine. If you need to install these first, contact your systems administrator for help.

Note: In order for the LispWorks IDE to run "out of the box", Motif must be installed on the target machine.

5.1.2 Disk requirements during installation

LispWorks requires about 46MB to install without documentation. Installing the documentation adds about 94MB (Professional / Enterprise / Academic) or 45MB (Personal) to this. A full installation of the Enterprise Edition with all documentation and loadable modules requires about 154MB.

The Professional / Enterprise / Academic documentation includes printable PDF format manuals. You may delete any of these that you do not need. They are available at **`www.lispworks.com/documentation`** in any case, and the same manuals are also available there in PostScript format.

5.1.3 Memory requirements at runtime

LispWorks 5.0 requires an absolute minimum of 32MB of user RAM in order to run. More RAM minimizes swapping and therefore improves performance. With 64MB RAM, LispWorks will perform well.

For information about the behavior of LispWorks near the limit of real memory, see "Memory requirements" on page 94.

5.2 License agreement

Before installing, you must read and agree to the license terms. To do this, mount the CD-ROM on your CD-ROM drive and locate the LispWorks for FreeBSD files in the **x86-freebsd** directory. Run the one of following scripts.

You must run this script as the same user that later performs the installation. In particular, if you are going to install LispWorks from the RPM file, you must run the license script while logged on as root.

- For the Professional and Enterprise Editions, run

  ```
  sh lwf-license.sh
  ```

  and enter “yes” if you agree to the license terms.

- For the Academic Edition, run

  ```
  sh lwfac-license.sh
  ```

  and enter “yes” if you agree to the license terms.

5.3 Software on the CD-ROM

LispWorks 5.0 for FreeBSD is supplied as a standard package file in the **x86-freebsd** directory on your CD-ROM.

5.3.1 Professional, Enterprise and Academic Edition distributions

All of the LispWorks modules are contained in a single package file. Your license key will control which modules can be used.

5.4 Installing LispWorks for FreeBSD

5.4.1 Main installation and patches

LispWorks Professional, Enterprise and Academic Editions are supplied as a standard software package file containing version 5.0. There may be a downloadable patch bundle which upgrades LispWorks to version 5.0.x. You need to complete the main installation before adding patches.

5.4.2 Information for Beta testers

Users of LispWorks 5.0 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 5.0.

See "Uninstalling LispWorks for FreeBSD" on page 41 for instructions.

5.4.3 Installation software package file

If you already have LispWorks 5.0 Beta installed, please uninstall it before installing this product. See Section 5.9, "Uninstalling LispWorks for FreeBSD".

5.4.3.1 Installation directories

By default LispWorks is installed in **`/usr/local/lib/LispWorks`** and a symbolic link to the executable is placed in **`/usr/local/bin/lispworks-5-0-0-x86-freebsd`**. However, the software package is relocatable, and the **`-p`** option can be used to allow the installation of LispWorks in a user-specified directory. The default prefix is **`/usr/local`**.

Note: the Academic Edition by default installs in **`/usr/local/lib/LispWorksAcademic`**. Do not attempt to to install different editions in the same location, since some filenames coincide and uninstallation may break.

5.4.3.2 Selecting the correct software package file

The LispWorks Professional/Enterprise software package file is called

```
lispworks-5.0.tgz
```

and can be found in the **`x86-freebsd`** directory of the LispWorks 5.0 CD-ROM.

The Academic Edition software package file is called

```
lispworks-academic-5.0.tgz
```

5.4.3.3 Installing LispWorks for FreeBSD

To install LispWorks, perform the following steps as root:

1. Locate the software package file.

2. Install or upgrade LispWorks in the standard way, for example:

   pkg_add lispworks-5.0.tgz

   This command installs LispWorks in **/usr/local/lib/LispWorks**. A command line of the form

   pkg_add -p <*directory*> lispworks-5.0.tgz

   installs LispWorks in <*directory*>.

The directory name must be an absolute pathname. Relative pathnames and pathnames including shell-expanded characters such as **.** and **~** do not work.

Note: LispWorks needs to be able find its library at runtime and therefore the LispWorks installation should not be moved around piecemeal. If you simply want to run LispWorks from somewhere more convenient, then consider adding a symbolic link.

See Section 5.6 for instructions on entering your license details.

5.4.3.4 Installation by non-root users

Non-root users should use the above installation procedure, but must specify the **-p** option to set a prefix a directory that is writable and also the -R option to prevent the package manager from attempting to update the package database.

Thus, a typical installation command for a non-root user is:

pkg_add -p *installation-directory* -R lispworks-5.0.tgz

All directory names must be absolute pathnames. Relative pathnames and pathnames including shell-expanded characters such as **.** and **~** do not work.

5.4.3.5 Installing Patches

After completing the main installation of the Professional, Enterprise or Academic Editions, ensure you install the latest patches from the package file available for download at **www.lispworks.com/downloads/patch-selection.html**. Patch installation instructions are in the README file accompanying the patch download.

5.5 LispWorks looks for a license key

If you installed the Professional, Enterprise or Academic Edition of LispWorks, the image looks for a valid license key. If you try to run these LispWorks Editions without a valid key, a message prints reporting that no valid key was found.

For instructions on entering your license key, see Section 5.6.1, “Entering the license data” below.

For more information about license keys, see Section 9.2, “License keys”.

5.6 Running LispWorks

The LispWorks executable is located in **`/usr/local/lib/LispWorks`** directory of the installation (assuming the default prefix of **`/usr/local`**) and should not be moved without being resaved because it needs to be able to locate the corresponding library directory on startup. There is also a symbolic link from the **`/usr/local/bin`** directory.

The LispWorks executable is named as shown here:.

`lispworks-5-0-0-x86-freebsd`	Professional or Enterprise Edition

When you run LispWorks, the Lisp Monitor and splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 93 for details if this does not happen.

5.6.1 Entering the license data

When you run the LispWorks Professional / Enterprise / Academic Edition for the first time, you will need to enter your license details. This should be done as follows (all on one line):

`lispworks-5-0-0-x86-freebsd --lwlicenseserial` ***`SERIALNUMBER`*** **`--lwlicensekey`** ***`LICENSEKEY`***

where *SERIALNUMBER* and *LICENSEKEY* are the strings supplied with LispWorks.

If you have received a media kit then your LispWorks license key is supplied on a label on the folder containing the CD-ROM. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

Contact **lisp-keys@lispworks.com** if you have problems with your LispWorks license key.

5.7 Configuring the image

If you installed the Professional, Enterprise or Academic Edition of LispWorks, you can now configure your LispWorks image to suit your needs and load the Professional or Enterprise Edition modules as necessary. For instructions, see Chapter 9, “Configuration Details on Linux and FreeBSD”.

5.8 Printable LispWorks documentation

In a default Professional / Enterprise / Academic installation, the **lib/5-0-0-0/manual/offline** directory contains PDF format versions of the manuals.

PostScript format versions of the manuals are also available for download.

5.9 Uninstalling LispWorks for FreeBSD

A software package containing LispWorks can be uninstalled in the usual way, for example by executing:

```
pkg_delete lispworks-5.0
```

If patches have been installed, then you will first need to uninstall that package, which will be named **lispworks-patches5.0**.

6

Installation on UNIX

6.1 Introduction

This chapter is a brief installation guide for UNIX LispWorks 5.0. Chapter 10 discusses installation and configuration in detail, but this chapter presents the minmum instructions necessary to get LispWorks up and running on your system. If you have difficulties installing LispWorks from these instructions, refer to the main guide, starting at Chapter 10, "Configuration Details on UNIX".

6.2 Extracting software from the CD-ROM

LispWorks 5.0 is supplied on a CD-ROM with the associated products CLIM 2.0, KnowledgeWorks, and LispWorks ORB. You will need root access while installing these products.

6.2.1 Finding out which CD-ROM files you need

The following table shows the platforms upon which LispWorks is supported:

Platform	Hardware code	OS code
HP PA (HP-UX 11x)	**hp-pa**	**hp-pa11**
Sun Sparc (32-bit, Solaris 2.8 & later)	**sparc**	**sparc-solaris**
Sun Sparc (64-bit, Solaris 2.8 & later)	**sparc64**	**sparc64-solaris**

Table 6.1 Platforms and associated codes

For HP PA (HP-UX 11x) you need the files named **lw50-hp-pa.tar** and **lwdoc50-unix.tar**.

For Sun Sparc (32-bit) you need the files named **lw50-sparc.tar** and **lwdoc50-unix.tar**.

For Sun Sparc (64-bit) you need the files named **lw50-sparc64.tar** and **lwdoc50-sparc64.tar**.

In each case the first archive contains the LispWorks image, libraries and examples and the layered products KnowledgeWorks, LispWorks ORB and CLIM. The second archive contains the documentation for Common Lisp, LispWorks and the layered products.

6.2.2 Unpacking the CD-ROM files

To unpack the CD-ROM files:

1. Mount the CD-ROM in your drive.
2. Search the subdirectories of the mount point to find the **tar** files.
3. Change directory to your installation directory (we recommend **/usr/lib/lispworks/**, which you may need to create) and decide which **tar** files you need.
4. Use the following command to unpack each **tar** file:

```
% tar -xof filename
```

The LispWorks image file can be found at top level in the installation directory, named according to the operating system, platform, and LispWorks version number, as follows:

lispworks-<*version number*>-<*OS code*>

Thus, an image named **lispworks-5-0-0-hp-pa11** would be a LispWorks 5.0 image for use on an HP PA machine running HP-UX 11.

6.3 Moving the LispWorks image and library

The LispWorks image must be able to find its library. The default library location is contained in the Lisp variable ***lispworks-directory***, but if that does not locate the library, LispWorks also can locate its library by a fallback mechanism which detects a numbered subdirectory **lib/5-0-0-0** alongside the image.

There are three distinct ways to arrange your LispWorks files. Choose 1, 2 or 3, of which 1 and 2 are the simplest options:

1. Put the LispWorks distribution in **/usr/lib/lispworks**. You will then have the LispWorks image at top-level in the **/usr/lib/lispworks** directory, and subdirectories **/usr/lib/lispworks/lib/5-0-0-0**.

   You can move the LispWorks image wherever your prefer, because the value of ***lispworks-directory*** in the supplied image is the pathname **#P"/usr/lib/lispworks/"**.

2. Keep the LispWorks installation intact, as unpacked from the archive supplied. You can move it, but only move the entire installation as a whole. Then LispWorks will find its library by the fallback mechanism mentioned above. In this case again you do not need to change ***lispworks-directory***.

   Note: this only works if you do not move the image away from the top-level of the installation directory.

3. Put the library elsewhere than **/usr/lib/lispworks/** (call it **/path/to/lwlibrary/**) and move the LispWorks image file away from the top-level of the installation directory.

In this case you need to take action to allow LispWorks to find its library. You should either make a symbolic link **/usr/lib/lispworks/lib**, or configure the LispWorks image with:

```
(setf *lispworks-directory* #P"/path/to/lwlibrary/")
```

See Section 6.5 below for more information about configuring LispWorks. You will need to install your license key first.

6.4 Obtaining and Installing your license keys

6.4.1 Keyfiles and the license server for HP PA and Sun Sparc (32-bit)

This section applies to platforms **hp-pa11** and **sparc-solaris** only.

LispWorks requires a license key in order to run. To make a key available to LispWorks, you must use either the keyfile system, or the License Server.

Most users use a keyfile. The License Server is more suitable for large sites with many LispWorks users.

6.4.1.1 If you are using the keyfile system

You will need a valid key, placed in a keyfile, for LispWorks to run. Note that keys and licenses issued for use with LispWorks version 4.x do not work for LispWorks 5.0.

To get a key for your copy of LispWorks, contact Lisp Support. You need to supply the machine ID. You can find this out by starting the LispWorks image up—the ID will be printed in the keyfile error message produced.

Send this information by e-mail to the following address:

lisp-keys@lispworks.com

Other queries should be sent to

lisp-support@lispworks.com

although please be sure to check Section 11.8, “Reporting bugs” for instructions before sending us a bug report. If you do not have e-mail access, you can contact Lisp Support by telephone or ordinary postal mail. Contact details are in Section 11.8.7, “Send the bug report”.

Once you have your key, put it in a file in one of the following locations:

- **keyfile.*hostname*** in the current working directory, where ***hostname*** is the name of the host machine on which LispWorks is to run
- **keyfile** in the current working directory
- **lib/5-0-0-0/config/keyfile.*hostname***, where ***hostname*** is the name of the host machine on which LispWorks is to run. The **lib** directory is expected by default to be located at **/usr/lib/lispworks/lib** (see Section 6.3 above)
- **lib/5-0-0-0/config/keyfile**, where the **lib** directory is as above.

If there is more than one key in the keyfile, make sure each one is on a separate line in the file and that there is no leading space before it.

For more details, see “How to obtain keys” on page 86.

6.4.1.2 If you are using the License Server

You will need to obtain permission codes from Lisp Support before you can get LispWorks up and running. Consult the *Lispworks Guide to the License Server*.

6.5 Configuring the LispWorks image

Now you can configure the LispWorks image to your taste. In the distribution directory **config** there are two files that have been preloaded into the LispWorks image:

- **config/configure.lisp**
- **config/a-dot-lispworks.lisp**

Take a look at the settings in **configure.lisp** to see if there is anything you want to change. In particular, you must change the value of ***lispworks-directory*** if you have chosen a location for the library which is different to that in the supplied image and moved the image away from the top-level of the installation directory.

If you already have a **.lispworks** personal initialization file in your home directory, examine the supplied example **a-dot-lispworks.lisp** file for new settings which you may wish to add. Otherwise, make a copy of

a-dot-lispworks.lisp in your home directory, naming it **.lispworks**. This file is loaded into LispWorks when you start it up, allowing you to make personal customizations to LispWorks not in the image your fellow users see.

6.5.1 Saving a configured image

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made any desired changes in **my-configuration.lisp** you can save a new LispWorks image, creating a local version.

1. Create a configuration and saving script **/tmp/config.lisp**, containing:

   ```
   (load-all-patches)
   (load "/tmp/my-configuration.lisp")
   (save-image "/usr/local/bin/lispworks")
   (quit)
   ```

2. Change directory to the top-level of the LispWorks installation directory, for example:

   ```
   % cd /usr/lib/lispworks
   ```

3. Start the supplied image using the configuration script as an **init** file. For example:

   ```
   % lispworks-5-0-0-sparc-solaris -siteinit - -init
   /tmp/config.lisp
   ```

If the image will not run at this stage, it is probably not finding a valid key. See “Obtaining and Installing your license keys” on page 46

The **siteinit.lisp** is also suppressed because this will be loaded automatically when you start the configured image. Saving the image takes some time.

You can now use the new image by starting it just as you did the supplied image. Saving a new image over the old one is not recommended. Use a unique name.

6.5.2 Testing the newly saved image

The following steps provide a basic test of your installation.

1. Change directory to **/tmp**.
2. Verify that your **DISPLAY** environment variable is correctly set and that your machine has permission to connect to the display.
3. Start up the new image.
4. Test the load-on-demand system:

   ```
   CL-USER 1 > (inspect 1)
   ```

   The inspector is a load-on-demand feature, so if the installation is correct you will see messages reporting that the inspector is being loaded.
5. Test the X interface:

   ```
   CL-USER 2 > (env:start-environment :display <display>)
   ```

   where **<display>** is the name of the machine running the X server, for example **"cantor:0"**.

6.6 Using the Documentation

Documentation in HTML and PDF formats is provided in a separate archive on the CD-ROM. If you want to access the documentation, you should unpack the appropriate archive named “Finding out which CD-ROM files you need” on page 44.

HTML documentation is installed in the **lib/5-0-0-0/manual/online** subdirectory of the LispWorks library, and can be accessed via the **Help** menu in the Common LispWorks IDE.

The PDF format manuals are installed in the **lib/5-0-0-0/manual/offline/pdf** subdirectory of the LispWorks library.

6.7 Using Layered Products on HP PA or Sun Sparc (32-bit)

To use each of LispWorks ORB, CLIM 2.0 and KnowledgeWorks you must obtain the required key and put in your keyfile. See “Keyfiles and the license server for HP PA and Sun Sparc (32-bit)” on page 46.

Then you need to load the layered product module. This is done by **`(require "corba")`** or **`(require "clim")`** or **`(require "kw")`**. You could consider configuring an image with the module pre-loaded, by using a **`config.lisp`** file similar to that in “Saving a configured image” on page 48.

Note: There is no additional licensing requirement for Common SQL on these platforms.

7

Configuration Details on Mac OS X

7.1 Introduction

This chapter explains how to get your LispWorks Professional, Enterprise or Academic Edition up and running, having already installed the files from the CD-ROM into an appropriate folder. If you have not done this, refer to Chapter 2, “Installation on Mac OS X”.

It is more useful to have an image customized to suit your particular environment and work needs. You can do this—setting useful pathnames, loading libraries, and so on—and then save the image to create another that will be configured as you require whenever you start it up.

This chapter covers the following topics:

- “License keys”
- “Configuring your LispWorks installation”
- “Saving and testing the configured image”
- “Initializing LispWorks”
- “Loading CLIM 2.0”
- “Loading Common SQL”
- “Common Prolog and KnowledgeWorks”

7.2 License keys

LispWorks is protected against unauthorized copying and use by a simple key mechanism. LispWorks will not start up until it finds a file containing a valid key.

The image looks for a valid license key in the following places, in order:

- in the current working directory (folder)
- in the directory containing the LispWorks executable
- in the **`Library/lib/5-0-0-0/config`** subdirectory of the LispWorks installation directory

When the file **`lwlicense`** is found, it must contain a valid key for the current machine. If you try to run LispWorks without a valid key, a message will be printed to the console reporting that no valid key was found.

7.3 Configuring your LispWorks installation

Once you have successfully installed and run LispWorks, you can configure it to suit your local conditions and needs, producing an image that is set up the way you want it to be every time you start it up.

7.3.1 Levels of configuration

There are two levels of configuration:

- configuring and resaving the image, thereby creating a new image that is exactly as you want it at startup
- configuring certain aspects of LispWorks as it starts up.

These two levels are available for good reason: while some configuration details may be of use to all LispWorks users on your machine (for instance, having a particular library built into the image where before it was only load-on-demand) others may be a matter of personal preference (for instance how many editor windows are allowed on-screen, or the colors of tool windows).

In the first case, you use edited copies of files in the **`config`** folder to achieve your aims.

In the second case, you make entries in your initialization file. This is a file read every time LispWorks starts up, and it can contain any valid Common Lisp code. (Most of the configurable settings in LispWorks can be controlled from Common Lisp.) By default the file is called **.lispworks** and is in your home directory. Your initialization file can be changed via **LispWorks > Preferences...** from the LispWorks IDE.

7.3.2 Configuring images for the different GUIs

If you have installed both the LispWorks images, for native Mac OS X and for X11/Motif, you will want to configure two images.

If necessary your Lisp configuration and initialization files can run code for one image or the other by conditionalization on the feature **:cocoa**. The native Mac OS X LispWorks image has **:cocoa** on ***features*** while the X11/Motif LispWorks image does not.

7.3.3 Configuration files available

There are four sample configuration files in LispWorks library containing settings you can change in order to configure images:

- **config/configure.lisp**
- **config/siteinit.lisp**
- **private-patches/load.lisp**
- **config/a-dot-lispworks.lisp**

config/configure.lisp is preloaded into the image before it is shipped. It contains settings governing fundamental issues like where to find the LispWorks runtime folder structure, and so on. You can override these settings in your saved image or in your initialization file. You should read through **configure.lisp** .

config/siteinit.lisp contains any forms that are appropriate to the whole site but which are to be loaded afresh each time the image is started. The sample **siteinit.lisp** file distributed with LispWorks contains only the form:

```
(load-all-patches)
```

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 7.4, below, and Section 7.5, “Initializing LispWorks” for further details.

private-patches/load.lisp is loaded by **load-all-patches**, and should contain forms to load any private (named) patches that Lisp Support might send you.

config/a-dot-lispworks.lisp is a sample personal initialization file. You might like to copy this into a file **~/.lispworks** in your home directory and edit it to create your own initialization file.

Both **configure.lisp** and **a-dot-lispworks.lisp** are preloaded into the image before it is shipped, so if you are happy with the settings in these files, you need not change them. See the example in Section 7.4, below, and Section 7.5, “Initializing LispWorks” for further details.

7.4 Saving and testing the configured image

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made the desired changes in **my-configuration.lisp** you can save a new LispWorks image. To do this, follow the instructions below.

1. Create a configuration and saving script **/tmp/save-config.lisp** containing:

```
(load-all-patches)
(load "/tmp/my-configuration.lisp")
#+:cocoa
(compile-file-if-needed
 (sys:example-file "configuration/macos-application-bundle")
 :load t)
(save-image #+:cocoa
            (write-macos-application-bundle
             "/Applications/LispWorks 5.0/My LispWorks.app")
            #-:cocoa
            "my-lispworks-motif")
(quit)
```

Note 1: The use of example code supplied with LispWorks which creates a Mac OS X application bundle. This code is in the example file **`examples/configuration/macos-application-bundle.lisp`**

Note 2: This will create a non-universal binary, containing only the architecture on which you call **`save-image`**.

2. Change directory to the directory containing the LispWorks image to configure. For the native Mac OS X LispWorks image (32-bit only):

   ```
   % cd "/Applications/LispWorks 5.0/LispWorks.app/Contents/MacOS"
   ```

   or for the X11 / Motif LispWorks image:

   ```
   % cd "/Applications/LispWorks 5.0"
   ```

3. Start the supplied image using the configuration script as an **`init`** file. For example enter one of the following commands (on one line of input):

   ```
   % lispworks-5-0-0-macos-universal -siteinit - -init
   /tmp/save-config.lisp
   ```

   or

   ```
   % lispworks-5-0-0-macos-universal-motif -siteinit - -init
   /tmp/save-config.lisp
   ```

   If the image will not run at this stage, it is probably not finding a valid key.

   Note that the command line also suppresses the **`siteinit`** because this will be loaded automatically when you start the configured image.

   Saving the image takes some time.

You can now use the new **`My LispWorks.app`** application bundle or the **`my-lispworks-motif`** image by starting it just as you did the supplied LispWorks. The supplied LispWorks is not required after the configuration process has been successfully completed.

Do not try to save a new image over an image that is currently running. Instead, save an image under a unique name, and then, if necessary, replace the new image with the old one after the call to **`save-image`** has returned.

Note: for the Academic Edition, the application folder and the image name differ slightly from the examples above.

7.4.1 Testing the newly saved image

You should now test the new LispWorks image. To test a configured LispWorks, do the following:

1. If you are using an X11 / Motif image, change directory to **/tmp**.
2. When using X11, verify that your **DISPLAY** environment variable is correctly set and that your machine has permission to connect to the display.
3. Start up the new image, by entering the path of the X11 / Motif executable or by double-clicking on the LispWorks icon in the Mac OS X Finder.

   The window-based environment should now initialize—during initialization a window displaying a copyright notice will appear on the screen.

   You may wish to work through some of the examples in the *LispWorks User Guide*, to further check that the configured image has been successfully built.
4. Test the load-on-demand system. In the Listener, type:

   ```
   CL-USER 1 > (inspect 1)
   ```

   Before information about the fixnum 1 is printed, the system should load the inspector from the **load-on-demand** Library directory.

7.4.2 Saving a non-windowing image

For some purposes such as scripting it is convenient to have a LispWorks image that does not start the graphical programming environment.

To save an image which does not automatically start the GUI, use a script as described in “Saving and testing the configured image” on page 54 but pass the **:environment** argument to **save-image**. For example:

```
(save-image "my-tty-lispworks" :environment nil)
```

7.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. The '**~**' denotes your home directory, indicated as **Home** in the Finder. The initialization file may contain any valid Lisp code.

You can load a different initialization file using the option **-init** in the command line, for example:

```
% "/Applications/LispWorks 5.0/LispWorks.app/Contents/MacOS/lispw
orks-5-0-0-macos-universal" -init my-lisp-init
```

(where **%** denotes the Unix shell prompt) would make LispWorks load **my-lisp-init.lisp** as the initialization file instead of that named by ***init-file-name***.

The loading of the siteinit file (located by default at **config/siteinit.lisp**) is similarly controlled by the **-siteinit** command line argument or ***site-init-file-name***.

You can start an image without loading any personal or site initialization file by passing a hyphen to the **-init** and **-siteinit** arguments instead of a filename:

```
% "/Applications/LispWorks 5.0/LispWorks.app/Contents/MacOS/lispw
orks-5-0-0-macos-universal" -init - -siteinit -
```

This starts the LispWorks image without loading any initialization file. It is often useful to start the image in this way when trying to repeat a suspected bug. You should always start the image without the default initialization files if you are intending to resave it.

In all cases, if the filename is present, and is not a hyphen, LispWorks tries to load it as a normal file by calling **load**. If the load fails, LispWorks prints an error report.

7.6 Loading CLIM 2.0

Load CLIM 2.0 into a LispWorks for X11/Motif image with

```
(require "clim")
```

and the CLIM demos with

```
(require "clim-demo")
```

A configuration file to save an image with CLIM 2.0 preloaded would look something like this:

```
(load-all-patches)
(require "clim")
(save-image "<destination>/clim-lispworks")
(quit)
```

To run the demo software, enter the following in a listener:

```
(require "clim-demo")
(clim-demo:start-demo)
```

Note: CLIM is not supported by the LispWorks native Mac OS X image and cannot be loaded into it.

Note: Do not attempt to load CLIM via the clim loader files in the clim distribution. This will cause CLIM patches to not be loaded. Use `(require "clim")`.

7.7 The Common SQL interface

The Common SQL interface requires ODBC or one of the supported database types listed in section "Supported Databases" of the *LispWorks User Guide*.

7.7.1 Loading Common SQL

To load Common SQL enter, for example:

```
(require "odbc")
```

or

```
(require "oracle")
```

Initialize the database type at runtime, for example:

```
(sql:initialize-database-type :database-type :odbc)
```

or

```
(sql:initialize-database-type :database-type :oracle)
```

See the *LispWorks User Guide* for further information.

7.7.2 Supported databases

Common SQL on Mac OS X has been tested with DBMS Postgres 7.2.1, MySQL 5.0.18, Oracle Instant Client 10.1.0.3, ODBC driver PSQLODBC development code, and IODBC as supplied with Mac OS X.

7.7.3 Special considerations when using Common SQL

7.7.3.1 Location of .odbc.ini

The current release of Mac OS X comes with an ODBC driver manager from IODBC, including a GUI interface. IODBC attempts to put the file **`.odbc.ini`** file in a non-standard location. This causes problems at least with the PSQLODBC driver for PostgreSQL, because PSQLODBC expects to find **`.odbc.ini`** in either the users's home directory or the current directory. There may be similar problems with other drivers. Therefore the file **`.odbc.ini`** should be placed in its standard place **`~/.odbc.ini`** . The IODBC driver manager looks there too, so it will work.

7.7.3.2 Errors using PSQLODBC

The PSQLODBC driver, when it does not find any of the Servername, Database or Username in **`.odbc.ini`**, returns the wrong error code. This tells the calling function that the user cancelled the login dialog.

Therefore, if Common SQL reports that the user cancelled when trying to connect, you need to check that you have got Servername, Database and Username, with the correct case, in the section for the datasource in the **`.odbc.ini`** file.

Note: Username may alternatively be given in the connect string.

7.7.3.3 PSQLODBC version

Common SQL was tested with the development version of psqlodbc (that is downloaded from CVS, with the version changed to 3. Contact Lisp Support if you need help using Common SQL with PSQLODBC.

7.7.3.4 Locating the Oracle, MySQL or PostgreSQL client libraries

For *database-type* **`:oracle`**, **`:mysql`** and **`:postgresql`**, if the client library is not installed in a standard place, its directory must be added to the environment variable DYLD_LIBRARY_PATH (see the OS manual entry for dyld).

7.8 Common Prolog and KnowledgeWorks

Common Prolog is bundled with KnowledgeWorks rather than with LispWorks. KnowledgeWorks is loaded by using:

```
(require "kw")
```

See the *KnowledgeWorks and Prolog User Guide* for further instructions.

8

Configuration Details on Windows

8.1 Introduction

This chapter explains how to get your LispWorks Professional, Enterprise or Academic Edition up and running, having already installed the files from the CD-ROM into an appropriate directory. If you have not done this, refer to Chapter 3, “Installation on Windows”.

It is more useful to have an image customized to suit your particular environment and work needs. You can do this—setting useful pathnames, loading libraries, and so on—and then save the image to create another that will be configured as you require whenever you start it up.

This chapter covers the following topics:

- “License keys”
- “Configuring your LispWorks installation”
- “Saving and testing the configured image”
- “Initializing LispWorks”
- “Loading CLIM 2.0”
- “The Common SQL interface”
- “Common Prolog and KnowledgeWorks”

8.2 License keys

LispWorks is protected against unauthorized copying and use by a simple key protection mechanism. LispWorks will not start up until it finds a valid key.

The image looks for a valid license key in the Windows registry.

If you try to run LispWorks without a valid key, it will prompt for a serial number and key.

8.3 Configuring your LispWorks installation

Once you have successfully installed and run LispWorks, you can configure it to suit your local conditions and needs, producing an image that is set up the way you want it to be every time you start it up.

8.3.1 Levels of configuration

There are two levels of configuration: configuring and resaving the image, thereby creating a new image that is exactly as you want it at startup, and configuring certain aspects of LispWorks as it starts up.

These two levels are available for good reason: while some configuration details may be of use to all LispWorks users on your site (for instance, having a particular library built in to the image where before it was only load-on-demand) others may be a matter of personal preference (for instance how many editor windows are allowed on-screen, or the colors of tool windows).

In the first case, you use edited copies of files in the **`config`** folder to achieve your aims.

In the second case, you make entries in your initialization file. This is a file read every time LispWorks starts up, and it can contain any valid Common Lisp code. (Most of the configurable settings in LispWorks can be controlled from Common Lisp.) Your initialization file can be changed via **`Tools > Global Preferences...`** in the Common LispWorks IDE.

8.3.2 Configuration files available

There are four sample configuration files in LispWorks library containing settings you can change in order to configure images:

- **config/configure.lisp**
- **config/siteinit.lisp**
- **private-patches/load.lisp**
- **config/a-dot-lispworks.lisp**

config/configure.lisp is preloaded into the image before it is shipped. It contains settings governing fundamental issues like where to find the LispWorks runtime folder structure, and so on. You can override these settings in your saved image or in your initialization file. You should read through **configure.lisp** .

config/siteinit.lisp contains any forms that are appropriate to the whole site but which are to be loaded afresh each time the image is started. The sample **siteinit.lisp** file distributed with LispWorks contains only the form:

```
(load-all-patches)
```

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 8.4, below, and Section 8.5, “Initializing LispWorks” for further details.

private-patches/load.lisp is loaded by **load-all-patches**, and should contain forms to load any private (named) patches that Lisp Support might send you.

config/a-dot-lispworks.lisp is a sample personal initialization file. You might like to copy this somewhere convenient and edit it to create your own initialization file.

Both **configure.lisp** and **a-dot-lispworks.lisp** are preloaded into the image before it is shipped, so if you are happy with the settings in these files, you need not change them.. See the example in Section 8.4, below, and Section 8.5, “Initializing LispWorks” for further details.

8.4 Saving and testing the configured image

Make a copy of **config\configure.lisp** called **C:\temp\my-configuration.lisp**. When you have made any desired changes in **my-configura-**

tion.lisp you can save a new LispWorks image. To do this, follow the instructions below.

1. Create a configuration and saving script **C:\temp\save-config.lisp**, containing:

```
(load-all-patches)
(load "C:\temp\my-configuration.lisp")
(save-image "my-lispworks")
(quit)
```

2. Change directory to the LispWorks installation directory, for example:

```
C:

cd C:\Program Files\LispWorks
```

3. Start the supplied image using the configuration script as an **init** file. For example:

```
C:\Program Files\LispWorks>lispworks-5-0-0-x86-win32.exe
-siteinit - -init C:\temp\save-config.lisp
```

If the image will not run at this stage, it is probably not finding a valid key.

Note that the command line also suppresses the **siteinit** because this will be loaded automatically when you start the configured image.

Saving the image takes some time.

You can now use the new **my-lispworks.exe** image from the Windows Explorer, or you may choose to add a shortcut. The supplied image is not required after the configuration process has been successfully completed.

Do not try to save a new image over an image that is currently running. Instead, save an image under a unique name, and then, if necessary, replace the new image with the old one after the call to **save-image** has returned.

Note: in the Academic Edition, the default installation location is

```
C:\Program Files\LispWorks Academic
```

and the image is

```
lispworks-academic-5-0-0-x86-win32.exe
```

8.4.1 Testing the newly saved image

You should now test the new LispWorks image. To test a configured version of LispWorks, do the following:

1. Start up the new image.

   The window-based environment should now initialize—during initialization a window displaying a copyright notice will appear on the screen.

   You may wish to work through some of the examples in the *LispWorks User Guide*, to further check that the configured image has been successfully built.

2. Test the load-on-demand system. In the Listener, type:

   ```
   CL-USER 1 > (inspect 1)
   ```

   Before information about the fixnum 1 is printed, the system should load the inspector from the **load-on-demand** directory.

8.4.2 Saving a non-windowing image

For some purposes such as scripting it is convenient to have a LispWorks image that does not start the graphical programming environment.

To save an image which does not automatically start the GUI, use a script as described in “Saving and testing the configured image” on page 63 but pass the **:environment** argument to **save-image**. For example:

```
(save-image "my-tty-lispworks" :environment nil)
```

8.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. You can use parse-namestring to see the expansion of this path. The file may contain any valid Lisp code.

You can load a different initialization file using the option **-init** in the command line, for example (all on one line):

```
C:\Program Files\LispWorks>lispworks-5-0-0-x86-win32.exe -init
my-lisp-init
```

would make LispWorks load **my-lisp-init.lisp** as the initialization file instead of that named by ***init-file-name***.

The loading of the siteinit file (located by default at **config\siteinit.lisp**) is similarly controlled by the **-siteinit** command line argument or ***site-init-file-name***.

You can start an image without loading any personal or site initialization file by passing a hyphen to the **-init** and **-siteinit** arguments instead of a file-name:

```
C:\Program Files\LispWorks>lispworks-5-0-0-x86-win32.exe -init -
-siteinit -
```

This starts the LispWorks image without loading any initialization file. It is often useful to start the image in this way when trying to repeat a suspected bug. You should always start the image without the default initialization files if you are intending to resave it.

In all cases, if the filename is present, and is not a hyphen, LispWorks tries to load it as a normal file by calling **load**. If the load fails, LispWorks prints an error report.

8.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 5.0 with

```
(require "clim")
```

and the CLIM demos with

```
(require "clim-demo")
```

rather than the clim loader files in the clim distribution (which were the entry points in LispWorks 3).

A configuration file to save an image with CLIM 2.0 preloaded would look something like this:

```
(load-all-patches)
(require "clim")
(save-image "<destination>/clim-lispworks")
(quit)
```

8.6.1 Running the CLIM demos

To run the demo software, enter the following in a listener:

```
(require "clim-demo")
(clim-demo:start-demo)
```

This displays a menu listing all the demos. Choose the demo you wish to see. More information about the demos is in section "The CLIM demos" of the *CommonLisp Interface Manager 2.0 User's Guide*

8.7 The Common SQL interface

The Common SQL interface requires ODBC or one of the supported database types listed in section "Supported databases" of the *LispWorks User Guide*.

8.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

```
(require "odbc")
```

and at runtime call:

```
(sql:initialize-database-type :database-type :odbc)
```

and then you can connect to any installed ODBC datasource.

To load the Common SQL interface to use MySQL, enter:

```
(require "mysql")
```

and at runtime call:

```
(sql:initialize-database-type :database-type :mysql)
```

See the *LispWorks User Guide* for further information.

8.8 Common Prolog and KnowledgeWorks

Common Prolog is bundled with KnowledgeWorks rather than with LispWorks. KnowledgeWorks is loaded by using:

```
(require "kw")
```

See the *KnowledgeWorks and Prolog User Guide* for further instructions.

9

Configuration Details on Linux and FreeBSD

9.1 Introduction

This chapter explains how to get your LispWorks Professional or Enterprise Edition up and running on Linux or FreeBSD, having already installed the files from the CD-ROM into an appropriate directory. If you have not done this, refer to Chapter 4, Installation on Linux or Chapter 5, Installation on FreeBSD.

It is more useful to have an image customized to suit your particular environment and work needs. You can do this—setting useful pathnames, loading libraries, and so on—and then save the image to create another that will be configured as you require whenever you start it up.

This chapter covers the following topics:

- “License keys”
- “Configuring your LispWorks installation”
- “Saving and testing the configured image”
- “Initializing LispWorks”
- “Loading CLIM 2.0”
- “The Common SQL interface”

- "Common Prolog and KnowledgeWorks"

9.2 License keys

LispWorks is protected against unauthorized copying and use by a simple key protection mechanism. LispWorks will not start up until it finds a file containing a valid key.

The image looks for a valid license key in the following places, in order:

- in the current working directory
- in the directory containing the LispWorks executable
- in the **lib/5-0-0-0/config** subdirectory of the LispWorks installation directory

When the file **lwlicense** is found, it must contain a valid key for the current machine. If you try to run LispWorks without a valid key, a message will be printed reporting that no valid key was found.

9.3 Configuring your LispWorks installation

Once you have successfully installed and run LispWorks, you can configure it to suit your local conditions and needs, producing an image that is set up the way you want it to be every time you start it up.

9.3.1 Levels of configuration

There are two levels of configuration: configuring and resaving the image, thereby creating a new image that is exactly as you want it at startup, and configuring certain aspects of LispWorks as it starts up.

These two levels are available for good reason: while some configuration details may be of use to all LispWorks users on your site (for instance, having a particular library built in to the image where before it was only load-on-demand) others may be a matter of personal preference (for instance how many editor windows are allowed on-screen, or the colors of tool windows).

In the first case, you use edited copies of files in the **config** directory to achieve your aims.

In the second case, you make entries in your initialization file. This is a file read every time LispWorks starts up, and it can contain any valid Common Lisp code. (Most of the configurable settings in LispWorks can be controlled from Common Lisp.) By default the file is called **.lispworks** and is in your home directory. Your initialization file can be changed via **Tools > Global Preferences...** in the Common LispWorks IDE.

9.3.2 Configuration files available

There are four sample configuration files in LispWorks library containing settings you can change in order to configure images:

- **config/configure.lisp**
- **config/siteinit.lisp**
- **private-patches/load.lisp**
- **config/a-dot-lispworks.lisp**

config/configure.lisp is preloaded into the image before it is shipped. It contains settings governing fundamental issues like where to find the LispWorks runtime folder structure, and so on. You can override these settings in your saved image or in your initialization file. You should read through **configure.lisp** .

config/siteinit.lisp contains any forms that are appropriate to the whole site but which are to be loaded afresh each time the image is started. The sample **siteinit.lisp** file distributed with LispWorks contains only the form:

```
(load-all-patches)
```

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 9.4, below, and Section 9.5, “Initializing LispWorks” for further details.

private-patches/load.lisp is loaded by **load-all-patches**, and should contain forms to load any private (named) patches that Lisp Support might send you.

config/a-dot-lispworks.lisp is a sample personal initialization file. You might like to copy this into a file **~/.lispworks** in your home directory and edit it to create your own initialization file.

Both **configure.lisp** and **a-dot-lispworks.lisp** are preloaded into the image before it is shipped, so if you are happy with the settings in these files, you need not change them. See the example in Section 9.4, below, and Section 9.5, “Initializing LispWorks” for further details.

9.4 Saving and testing the configured image

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made any desired changes in **my-configuration.lisp** you can save a new LispWorks image. To do this, follow the instructions below.

1. Create a configuration and saving script **/tmp/save-config.lisp**, containing:

   ```
   (load-all-patches)
   (load "/tmp/my-configuration.lisp")
   (save-image "my-lispworks")
   (quit)
   ```

2. Change directory to the LispWorks installation directory, for example:

   ```
   % cd /usr/local/lib/LispWorks
   ```

3. Start the supplied image using the configuration script as an **init** file. For example:

   ```
   % lispworks-5-0-0-x86-linux -siteinit - -init
   /tmp/save-config.lisp
   ```

If the image will not run at this stage, it is probably not finding a valid key.

Note that the command line also suppresses the **siteinit** because this will be loaded automatically when you start the configured image.

Saving the image takes some time.

You can now use the new **my-lispworks** image by starting it just as you did the supplied image. The supplied image is not required after the configuration process has been successfully completed.

Do not try to save a new image over an image that is currently running. Instead, save an image under a unique name, and then, if necessary, replace the new image with the old one after the call to **save-image** has returned.

9.4.1 Testing the newly saved image

You should now test the new LispWorks image. To test a configured version of LispWorks, do the following:

1. Change directory to **/tmp**.
2. Verify that your **DISPLAY** environment variable is correctly set and that your machine has permission to connect to the display.
3. Start up the new image.

   The window-based environment should now initialize—during initialization a window displaying a copyright notice will appear on the screen.

   You may wish to work through some of the examples in the *LispWorks User Guide*, to further check that the configured image has been successfully built.
4. Test the **load-on-demand** system. In the Listener, type:

   ```
   CL-USER 1 > (inspect 1)
   ```

   Before information about the fixnum 1 is printed, the system should load the inspector from the **load-on-demand** directory.

9.4.2 Saving a non-windowing image

For some purposes such as scripting it is convenient to have a LispWorks image that does not start the graphical programming environment.

To save an image which does not automatically start the GUI, use a script as described in “Saving and testing the configured image” on page 72 but pass the **:environment** argument to **save-image**. For example:

```
(save-image "my-tty-lispworks" :environment nil)
```

9.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. ~ denotes your home directory. The file may contain any valid Lisp code.

You can load a different initialization file using the option **-init** in the command line, for example:

```
% lispworks-5-0-0-x86-linux -init my-lisp-init
```

would make LispWorks load **my-lisp-init.lisp** as the initialization file instead of that named by ***init-file-name***.

The loading of the siteinit file (located by default at **config/siteinit.lisp**) is similarly controlled by the **-siteinit** command line argument or ***site-init-file-name***.

You can start an image without loading any personal or site initialization file by passing a hyphen to the **-init** and **-siteinit** arguments instead of a filename:

```
% lispworks-5-0-0-x86-linux -init - -siteinit -
```

This starts the LispWorks image without loading any initialization file. It is often useful to start the image in this way when trying to repeat a suspected bug. You should always start the image without the default initialization files if you are intending to resave it.

In all cases, if the filename is present, and is not a hyphen, LispWorks tries to load it as a normal file by calling **load**. If the load fails, LispWorks prints an error report.

9.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 5.0 with

```
(require "clim")
```

and the CLIM demos with

```
(require "clim-demo")
```

rather than the clim loader files in the clim distribution (which were the entry points in LispWorks 3).

A configuration file to save an image with CLIM 2.0 preloaded would look something like this:

```
(load-all-patches)
(require "clim")
(save-image "<destination>/clim-lispworks")
(quit)
```

9.6.1 Running the CLIM demos

To run the demo software, enter the following in a listener:

```
(require "clim-demo")
(clim-demo:start-demo)
```

This displays a menu listing all the demos. Choose the demo you wish to see. More information about the demos is in section "The CLIM demos" of the *CommonLisp Interface Manager 2.0 User's Guide*

9.7 The Common SQL interface

The Common SQL interface requires ODBC or one of the supported database types listed in section "Supported databases" of the *LispWorks User Guide*.

9.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

```
(require "odbc")
```

and at runtime call:

```
(sql:initialize-database-type :database-type :odbc)
```

and then you can connect to any installed ODBC datasource.

To load the Common SQL interface to use MySQL, enter:

```
(require "mysql")
```

and at runtime call:

```
(sql:initialize-database-type :database-type :mysql)
```

See the *LispWorks User Guide* for further information.

9.8 Common Prolog and KnowledgeWorks

Common Prolog is bundled with KnowledgeWorks rather than with LispWorks. KnowledgeWorks is loaded by using:

```
(require "kw")
```

See the *KnowledgeWorks and Prolog User Guide* for further instructions.

10

Configuration Details on UNIX

10.1 Memory Requirements

This section discusses LispWorks 5.0 and its associated products' memory (RAM and hard disk) requirements. Make sure you have sufficient of both to install the products you have bought before moving on to unpacking them from the CD-ROM.

10.1.1 Disk requirements

The LispWorks software requires up to 53MB of diskspace, depending on the platform.

Installing the documentation adds up to 66MB to this. You can delete some of these files if you wish, for example you might not need the PDF manuals in **lib/5-0-0-0/manual/offline/pdf** (28Mb). You can download these PDF format manuals from **www.lispworks.com/documentation** at any time, and the same manuals are also available there in PostScript format. However, note that the **Help** menu commands will not work if you corrupt the **lib/5-0-0-0/manual/online** directory of the LispWorks library.

10.1.2 Memory requirements at runtime

LispWorks 5.0 requires an absolute minimum of 32MB of user RAM in order to run. More RAM minimizes swapping and therefore improves performance. With 64MB RAM, LispWorks will perform well.

For information about the behavior of LispWorks near the limit of real memory, see “Memory requirements” on page 94.

10.2 Software Requirements

The LispWorks for UNIX GUI requires X11 release 5 or above, and Motif version 2.

10.3 Unpacking the CD-ROM

This section explains the organization of the LispWorks 5.0 CD-ROM and how to unpack the files for the Lisp products you have bought. There are sections explaining the process for each supported platform.

10.3.1 The LispWorks 5.0 CD-ROM

The CD-ROM contains images for LispWorks 5.0 and associated products on your platform or platforms.

10.3.1.1 CD-ROM format

The files on the CD-ROM were created with the UNIX **`tar`** command.

10.3.2 Unpacking LispWorks products

There are two basic steps in unpacking a LispWorks product from the CD-ROM:

1. Mount the CD-ROM so that it can be accessed as part of your UNIX file-system
2. Extract the product files from the **`tar`** file containing them.

To do the latter, you can take a copy of the appropriate **`tar`** file and put it where you want to install the product, then extract the product files from that **`tar`** file, or alternatively, extract the product files direct from the **`tar`** file on CD-ROM.

10.3.3 Mounting the CD-ROM

Before you can access the files on the CD-ROM, it has to be mounted onto your UNIX filesystem. You may need root access on your machine to do this.

On some platforms, the CD-ROM will be mounted automatically when you place it in the drive. On most, however, you will have to run a mounting program to mount it. You may also have to create a directory on your machine to serve as the mount point. (The mount point is the point in your filesystem at which the CD-ROM directory structure will be found.)

When you have mounted the CD-ROM and can see the **`tar`** files on your UNIX filesystem, you are ready to unpack them. Once you are finished with the **`tar`** files on the CD-ROM, you can remove it from your drive, but *only* after you have performed an "unmount" operation.

When unmounting it is necessary that no process has the CD-ROM mount point as the current directory, and again, root access is necessary. Pushing the eject button on the drive may not do anything until the volume has been unmounted.

The basic syntax of the mounting and unmounting operations on each supported platform is given in each of the platform-specific sections below.

10.3.3.1 HP UX (HP Precision Architecture)

To mount:

```
mount -F cdfs -o cdcase /dev/dsk/c0t4d0 /mount-point
```

where *mount-point* is the directory over which you wish to mount the CD-ROM. The device designation **`/dev/dsk/c0t4d0`** may vary.

To unmount:

```
umount /dev/dsk/c0t4d0
```

Again, use the appropriate device designation for your hardware.

10.3.3.2 Solaris (Sun Sparc)

To mount: Solaris provides an automounting daemon. Place the CD-ROM in the drive and it will be automatically mounted to:

```
/cdrom/lw_50/
```

To unmount:

```
umount /cdrom/lw_50/
```

10.3.4 Unpacking the TAR files

Once the CD-ROM is mounted, you can begin to unpack the **tar** files for the products you have purchased. You will need root access to do this.

10.3.4.1 Considerations to be made before extracting product files

When you extract files made with the **tar** command, they are written into the current directory, and if there are any directories packed up in the tar file, they will be written to the current directory too. For this reason it is best to **cd** to the correct directory before extracting anything.

Consider who is going to use LispWorks before you decide where to put the extracted files. Once installed and configured, the executable Lisp image should be somewhere in the UNIX file system likely to be on its users' search path. A suitable place might be **/usr/local/bin/lispworks**.

The run time directory structure (basically, everything except the image file) should be somewhere publicly readable: **/usr/lib/lispworks**, by default. If there is not enough room in any of the normal publicly accessible locations, you could put a symbolic link there pointing to an installation directory in a partition with more disk space.

10.3.4.2 Keeping your old LispWorks installation

You can install LispWorks 5.0 in the same directory as previous versions of LispWorks such as LispWorks 4.4. This is because all the 5.0 files are stored in a subdirectory called **lib/5-0-0-0**.

You must recompile all your code with the LispWorks 5.0 compiler.

Binaries produced by **compile-file** in previous versions of LispWorks such as LispWorks 4.4 do not load into a LispWorks 5.0 image.

10.3.4.3 How to extract the product files from the tar container files

To extract the product files from the **tar** container files, the basic form of the call to **tar** is:

```
tar -xof /mount-point/filename
```

The flag **x** means extract files from **tar**-formatted data, and **f** specifies that the source of the data will be a file.

mount-point is the point in the UNIX filesystem at which the CD-ROM is mounted, while *filename* is the name of the **tar** file containing the product files.

For example, to extract the files for LispWorks (32-bit) on Solaris, with the CD-ROM mounted at **/cdrom/lw_50/**, you would type

```
tar -xof /cdrom/lw_50/lw50-sparc.tar
```

10.3.4.4 HP UX (HP Precision Architecture)

The files you need to unpack for LispWorks on HP UX are **lw50-hp-pa.tar** and **lwdoc50-unix.tar**.

The LispWorks image is

```
./lispworks-5-0-0-hp-pa11
```

10.3.4.5 Solaris (LispWorks 32-bit)

The files you need to unpack for LispWorks (32-bit) on Solaris are **lw50-sparc.tar** and **lwdoc50-unix.tar**.

The LispWorks image is

```
./lispworks-5-0-0-sparc-solaris
```

10.3.4.6 Solaris (LispWorks 64-bit)

The files you need to unpack for LispWorks (64-bit) on Solaris are **`lw50-sparc64.tar`** and **`lwdoc50-sparc64.tar`**.

The LispWorks image is

```
./lispworks-5-0-0-sparc64-solaris
```

10.4 Installing LispWorks

This section explains how to get LispWorks up and running, having already unpacked the **`tar`** files from the CD-ROM into an appropriate directory. If you have not done this, refer to Section 10.3, “Unpacking the CD-ROM”.

10.4.1 Introduction

We assume that the X Window System and a suitable version of Motif are installed on your machine. If you need to install it first, contact your systems administrator for help.

There are several images on the CD-ROM, one for each of the architectures LispWorks can run on. These images have been saved in a generic form, with popular libraries already present, less popular libraries left to be demand-loaded, and with internal pathname variables set to values that will probably need adjusting to suit your machine better.

It is more useful to have an image customized to suit your particular environment and work needs. You can do this—setting useful pathnames, loading libraries, and so on—and then save the image to create another that will be configured as you require whenever you start it up.

This section covers the following topics:

- Components of the LispWorks distribution
- Printing copies of the LispWorks documentation
- Keyfiles and how to obtain them
- Configuring your LispWorks installation
- Saving and testing a configured image

10.4.2 Components of the LispWorks distribution

For the purposes of installation the LispWorks system can be thought of as two discrete components: the basic executable Lisp image and the directories holding files consulted at runtime.

10.4.2.1 The LispWorks image

The supplied LispWorks image is named according to the operating system and platform for which it is built, and the LispWorks version number. The format is:

lispworks-<*version number*>-<*OS code*>

Thus, an image named **lispworks-5-0-0-sparc-solaris** is the LispWorks 5.0 image for use on Sun Sparc (32-bit) Solaris machines.

As noted in Section 10.3.4.1 on page 80, once installed, the basic executable Lisp image can be placed somewhere in the UNIX file system likely to be on its users' search path. A suitable place might be **/usr/local/bin/lispworks**.

10.4.2.2 The LispWorks library

The runtime directory structure (basically, everything except the image file) should be somewhere publicly readable: **/usr/lib/lispworks**, by default. If there is not enough room in any of the normal publicly accessible locations, you could put a symbolic link there pointing to the installation directory in a partition with more disk space. The installation directory must contain a subdirectory called **lib/5-0-0-0/**.

Among the directories on this subdirectory are the following:

- **config** — various files that can be adjusted in order to customize the image (see Section 10.4.6 on page 87).
- **app-defaults** — X/Motif resources for LispWorks and the Lisp Monitor.
- **postscript** — printer descriptions for the CAPI printing interface.
- **etc** — the executable for the Lisp Monitor.

- **load-on-demand** — Lisp library code that is loaded into a running LispWorks system as and when required.
- **patches** — numbered patches to LispWorks and layered products.
- **private-patches** — the location to place private (named) patches that Lisp support may send to you.
- **hqn_ls** — executables for the License server. You do not need these if all your products are licensed by keyfiles.
- **examples** — directories containing various code examples, including most of the code printed in the user documentation.
- **translations** — the place for logical pathname translations settings
- **src** — source code supplied with LispWorks

The following directory also resides here, but comes from the documentation archive:

- **manual** — has two subdirectories: **online** and **offline**. The directory **online** contains the online documentation. The directory **offline/pdf** contains the PDF versions of the complete LispWorks manual set.

By default, all these directories are assumed to reside beneath **/usr/lib/lispworks/lib/5-0-0-0/**, although you may place the **lib** directory somewhere else.

For products which support the License Server, there is also a subdirectory of the installation directory called **hqn_ls**.

10.4.3 Printing copies of the LispWorks documentation

LispWorks documentation is not supplied in printed form. If you own a LispWorks license, you may print extra copies of the manuals found in the LispWorks distribution, provided that each copy includes the complete copyright notice.

The **offline/pdf** directory contains PDF versions of each manual.

10.4.4 Keyfiles and how to obtain them

This section applies only to HP PA and Sun Sparc (32-bit).

LispWorks is protected against unauthorized copying and use by a simple key protection mechanism. LispWorks will not start up until it finds a file containing a valid key.

10.4.4.1 Where LispWorks looks for keyfiles

The image looks for a valid keyfile in the following places, in order:

- **keyfile.***hostname* in the current working directory, where *hostname* is the name of the host.
- **keyfile** in the current working directory, where *hostname* is the name of the host.
- **config/keyfile.***hostname*, where *hostname* is the name of the host on which the image is to execute. The **config** directory is expected by default to be located at **/usr/lib/lispworks/lib/5-0-0-0/config** (see "If you are using the keyfile system" on page 46.
- **config/keyfile**, where the **config** directory is as above.

The directory **config** is an indirect subdirectory of the directory specified by the LispWorks variable ***lispworks-directory***. Note that until you have configured and saved your image, as described later in this section, this variable is set to **/usr/lib/lispworks**. When starting the generic image, you must therefore ensure that the keyfile is either in your current directory or in **/usr/lib/lispworks/lib/5-0-0-0/config**.

If you try to run LispWorks without a valid key, a message will be printed reporting that no valid key was found.

10.4.4.2 The contents of a keyfile

Keyfiles contain one or more keys. A key is a sequence of 28 ASCII upper case letters and digits between 2 and 9, inclusive.

Each key should be placed on a separate line in the file. There should be no leading white space on a line before the start of a key. Characters after the key but on the same line as it are ignored, so may be used for comments. Indeed it is helpful to comment each line with the name of the product that key enables.

Key files for more than one host can exist in the same keyfile.

A single key allows you to use a particular major version of LispWorks (in this case 5), on one host machine, until the expiry date of one license, where relevant. To run LispWorks on a different machine you will need another key.

Delivery, KnowledgeWorks, LispWorks ORB and CLIM 2.0 each need their own keys.

10.4.4.3 How to obtain keys

To obtain your keys, contact Lisp Support.

You can get your key by phone, fax or email. Every key is unique: in order to generate keys, we need to know the unique ID of the machine on which you intend to run LispWorks.

To find out your machine's ID, try to start up the LispWorks image. LispWorks spots that there is no valid key available, and prints a message saying so, along with the ID you need to let us know. In any case, Lisp Support will be able to provide assistance in determining the identifier of a specific machine. We will also retain a copy of the key supplied.

Send email containing the message printed to **`lisp-keys@lispworks.com`**. Or contact Lisp Support as described in "Reporting bugs" on page 106.

Once you have the key, write it into a file in one of the places listed in Section 10.4.4.1, and start up the LispWorks image.

10.4.5 The License Server

This section applies only to HP PA and Sun Sparc (32-bit)..

If you prefer, you can run LispWorks using the License Server instead of the keyfile system. This system will control license allocation across your LAN, and you may find it more convenient.

See the *Lispworks Guide to the License Server* for full details.

As with the keyfile system, you will need to contact Lisp Support to obtain the necessary permissions.

10.4.6 Configuring your LispWorks installation

Once you have successfully installed and run LispWorks, you can configure it to suit your local conditions and needs, producing an image that is set up the way you want it to be every time you start it up.

There are two levels of configuration: configuring and resaving the image, thereby creating a new image that is exactly as you want it at startup, and configuring certain aspects of LispWorks as it starts up.

These two levels are available for good reason: while some configuration details may be of use to all LispWorks users on your site (for instance, having a particular library built in to the image where before it was only load-on-demand) others may be a matter of personal preference (for instance how many editor windows are allowed on-screen, or the colors of tool windows).

In the first case, you alter the global LispWorks image and global settings files in the **config** directory to achieve your aims.

In the second case, you make entries in a file in your home directory called **.lispworks**. This is a file read every time LispWorks starts up, and it can contain any valid Common Lisp code. (Most of the configurable settings in LispWorks can be controlled from Common Lisp.)

10.4.6.1 Multiple-platform installations

You can install copies of LispWorks for more than one platform in the same directory hierarchy. All platform-specific files are supplied with platform-specific names.

10.4.6.2 Configuration files available

There are four files in the LispWorks library containing settings you can change in order to configure images:

- **config/configure.lisp**
- **config/siteinit.lisp**
- **private-patches/load.lisp**
- **config/a-dot-lispworks.lisp**

config/configure.lisp contains settings governing fundamental issues like where to find the LispWorks runtime directory structure, and so on. You should read through **configure.lisp** and check that you are happy with all the settings therein. The most common change required is to ***lispworks-directory***, which points to the root of the installation hierarchy.

config/siteinit.lisp contains any forms that are appropriate to the whole site but which are to be loaded afresh each time the image is started. The sample **siteinit** file distributed with LispWorks contains only the form:

```
(load-all-patches)
```

private-patches/load.lisp is loaded by **load-all-patches**, and should contain forms to load any private (named) patches that Lisp Support might send you.

config/a-dot-lispworks.lisp is a sample **.lispworks** file. You might like to copy this into your home directory and use it as a basis for your own **.lispworks** file.

Both **configure.lisp** and **a-dot-lispworks.lisp** are preloaded into the image before it is shipped, so if you are happy with the settings in these files, you need not change them.

On startup, the image loads **siteinit.lisp** and your **.lispworks** file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 10.4.7 below, and see also Section 10.5, “Initializing LispWorks” for further details.

10.4.7 Saving and testing the configured image

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made any desired changes in **my-configuration.lisp** you can save a new LispWorks image. To do this, follow the instructions below.

1. Change directory to the installation directory, for example:

   ```
   unix% cd /usr/lib/lispworks
   ```

2. Start the supplied image, without loading any initialization files. For example:

   ```
   unix% lispworks-5-0-0-sparc-solaris -init - -siteinit -
   ```

   If the image will not run at this stage, it is probably not finding a valid key. See “Keyfiles and how to obtain them” on page 84.

3. Wait for the prompt. Load your local configuration file:

   ```
   CL-USER 1 > (load "/tmp/my-configuration.lisp")
   ```

   Now load all current patches:

   ```
   CL-USER 2 > (load-all-patches)
   ```

4. Save the new version of the image. For example:

   ```
   CL-USER 3 > (save-image "/usr/local/bin/lispworks")
   ```

Saving the image takes some time.

You can now use the new image by starting it just as you did the generic image. The generic image will not be required after the installation process has been completed successfully.

Do not try to save a new image over an image that is currently running. Instead, save an image under a unique name, and then, if necessary, replace the new image with the old one after the call to **`save-image`** has returned.

10.4.7.1 Testing the newly saved image

You should now test the new LispWorks image. To test a configured version of LispWorks, do the following:

1. Change directory out of the installation directory.
2. Run the new image.
3. Test the load-on-demand system. Type:

   ```
   CL-USER 1 > (inspect 1)
   ```

   Before information about the fixnum 1 is printed, the system should load the inspector from the **`load-on-demand`** directory.

4. Next, test the ability of the system to interface to a local X server. If necessary, start an X server either on the local machine or on a machine networked to it. Type:

```
CL-USER 2 > (env:start-environment :display "serverhostname")
```

Where *serverhostname* is the name of the machine running the X server. The window-based environment should now initialize—during initialization an X window displaying a copyright notice will appear on the screen.

You can work through some of the examples in the *LispWorks User Guide* to check further that the configured image has successfully built.

10.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is "**~/.lispworks**" by default. The file may contain any valid Lisp code.

You can load a different initialization file using the option **-init** in the command line, for example:

```
unix% lispworks -init my-lisp-init
```

would make LispWorks load my-lisp-init.lisp as the initialization file instead of that named by ***init-file-name***.

Alternatively, an initialization file may be specified by setting the UNIX environment variable **LW_INIT**. If set, the specified file will be used instead of that named by ***init-file-name***.

The loading of the siteinit file (located by default at **config/siteinit.lisp**) may similarly be controlled either by the **-siteinit** command line argument, or the **LW_SITE_INIT** variable and ***site-init-file-name***.

You can start an image without loading any personal or site initialization file by passing a hyphen to the **-init** and **-siteinit** arguments instead of a filename:

```
unix% lispworks -init - -siteinit -
```

This starts the LispWorks image without loading any initialization file. It is often useful to start the image in this way when trying to repeat a suspected

bug. You should always start the image without initialization if you are intending to resave it.

In all cases, if the filename is non-nil, and is not a hyphen, LispWorks tries to load it as a normal file by calling **load**. If the load fails, LispWorks prints an error report.

11

Troubleshooting, Patches and Reporting Bugs

This chapter discusses other issues that arise when installing and configuring LispWorks. It provides solutions for possible problems you may encounter, and it discusses the patch mechanism and the procedure for reporting bugs.

11.1 Troubleshooting

This section describes some of the most common problems that can occur on any platform during installation or configuration.

11.1.1 License key errors in the Professional and Enterprise Editions

LispWorks looks for a valid license key when it is started up. If a problem occurs at this point, LispWorks exits

These are the possible problems:

- LispWorks cannot find or read the key.
- The key is incorrect.
- Your license has expired, making the key no longer valid.

On Linux and FreeBSD, this is also a possible cause of the problem:

- The machine name has changed since LispWorks was installed.

On Mac OS X, Linux and FreeBSD, the key is expected to be stored in a keyfile, and an appropriate error message is printed for each case.

On Windows, the key is expected to be stored in the Windows registry.

11.1.2 Failure of the load-on-demand system

Module files are in the modules directory **lib/5-0-0-0/load-on-demand** under ***lispworks-directory***.

If loading files on demand fails to work correctly, check that the modules directory is present. If it is not, perhaps your LispWorks installation is corrupted.

Do not remove any files from the modules directory unless you are really certain they will never be required.

The supplied image contains a trigger which causes ***lispworks-directory*** to be set on startup and hence you should not need to change its value. Subsequently saved images do not have this trigger.

11.1.3 Memory requirements

LispWorks 5.0 requires a minimum amount of memory as described in “Memory requirements at runtime” on page 23.

The amount of swap space required to run LispWorks efficiently obviously depends on the base image size and the amount of application code loaded into the image.

To run the full LispWorks system, with its GUI, you will need around 20MB of swap space for the image alone, and whatever else is necessary to accommodate your application.

When running a large image, you may occasionally see

```
<**> Failed to enlarge memory
```

printed to the standard output.

The message means that the LispWorks image attempted to expand one of the GC generations, but there was not enough swap space to accommodate the

resulting growth in image size. When this happens, the garbage collector is invoked, and it will usually manage to free the required space.

Occasionally, however, continued demand for additional memory will end up exhausting resources. You will then see the message above repeatedly, and there will be little or no other activity apparent in the image. At this point you should restart the image, or increase swap space.

11.2 Troubleshooting on Mac OS X

This section describes some of the most common problems that can occur during installation or configuration of LispWorks for Macintosh.

If you're using the LispWorks image with the X11 / Motif GUI, see also Section 11.6, "Troubleshooting on X11 / Motif" below for issues specific to X11 / Motif.

11.2.1 Default installation requires administrator on Mac OS X

To install LispWorks in the default installation location under **`/Applications`** you must log on as an administrator. So it is usually best to run **`LispWorks_Installer`** as an administrator - the account you created when setting up your Macintosh is an administrator, for instance.

However, a non-administrator may install LispWorks elsewhere.

11.2.2 Failure to start when disconnected from the Internet

By default MacOS X machines have different names when connected and when not connected. After changing the connection state, the 64-bit LispWorks license check will fail, because the data is encoded with the machine name.

The machine name is configured by the line

```
HOSTNAME=-AUTOMATIC-
```

in **`/etc/hostconfig`**.

The recommended fix is to edit **`/etc/hostconfig`** to give your machine a fixed hostname, then reset the license file if necessary by running in Terminal.app:

```
$ ./lispworks-5-0-0-darwin64-motif --lwlicenseserial
SERIALNUMBER --lwlicensekey LICENSEKEY
```

where *SERIALNUMBER* and *LICENSEKEY* are the strings supplied with LispWorks. Then the LispWorks license check will always lookup the expected hostname.

Note: this section does not apply to 32-bit LispWorks for Macintosh - see “Licensing problems with Macintosh hostnames fixed” on page 139.

11.2.3 Text displayed incorrectly in the editor on Mac OS X

The LispWorks editor currently relies on integral font sizes. Some Mac OS X fonts have non-integral size and will be displayed incorrectly in the Editor and Listener tools and other uses of **capi:editor-pane**.

The solution is to use a font with integral size. The following are known to work: Monaco 10, Monaco 15, Monaco 20.

Select the font in an Editor or Listener tool by **Window > Window Preferences... > Font**.

11.3 Troubleshooting on Linux

This section describes some of the most common problems that can occur during installation or configuration of LispWorks for Linux.

See also “Troubleshooting on X11 / Motif” on page 101 below for issues specific to X11 / Motif.

11.3.1 Processes hanging

Some versions of Linux have a broken pthreads library. To workaround this set the environment variable LD_ASSUME_KERNEL=2.4.19 before running LispWorks.

LD_ASSUME_KERNEL allows using older versions of pthreads, some of which do not work.

LispWorks 5.0 supports kernel versions 2.4.20 and later.

11.3.2 RPM_INSTALL_PREFIX not set

On Linux, during installation of CLIM, Common SQL, LispWorks ORB or KnowledgeWorks from a secondary rpm file you may see a message similar to this:

```
# rpm --install tmp/lispworks-clim-5.0-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-5.0-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-5.0-1
#
```

This is only a problem when LispWorks itself was installed in a non-default location (that is, using the **`--prefix`** RPM option). You would then want to supply that same **`--prefix`** value when installing the secondary rpm. A bug in RPM means that a required environment variable **`RPM_INSTALL_PREFIX`** is not set automically to the supplied value. We have seen this bug in RPM version 4.2, as distributed with RedHat 8 and 9.

The workaround is to set this environment variable explicitly before installing the secondary rpm. For example, if LispWorks was installed like this:

```
rpm --install --prefix /usr/lisp lispworks-5.0-1.i386.rpm
```

then you would add CLIM like this (in C shell):

```
setenv RPM_INSTALL_PREFIX /usr/lisp
rpm --install --prefix /usr/lisp lispworks-clim-5.0-1.i386.rpm
```

11.3.3 Choosing between Motif and Lesstif on Linux

To support the detection of your preferred library, there is a configuration variable **`x-utils::*use-motif-library*`** and also a configuration file which the variable can refer to.

On startup, if the **`:config-file`** is specified in **`*use-motif-library*`**, then it is processed as described below to reset **`*use-motif-library*`**. The **`:detect-version`** and **`:prefer-version`** values are then processed as described below.

The value of **`*use-motif-library*`** is a plist with the following keys:

:detect-version

A list of lists, mapping library names to Xm versions. The Xm version should be either **:lesstif** (for Lesstif which supports Xm 2.1) or **:motif** (for OpenMotif which supports Xm 2.2). Note that Lesstif's Xm 1.x libraries are not supported.

:prefer-version

Either **:lesstif** or **:motif** indicating the preferred Xm version. If more than one of the *detect-version* files is found then the *prefer-version* selects the version to use. If none of the *detect-version* files is found then the *prefer-version* is used as the fallback.

:config-file

A string naming a configuration file. If the key is present, the file is opened, a single Lisp form is read and ***use-motif-library*** is set to this form. This causes the settings in the file to override those in the image. If the file name is not absolute then it is obtained from the **config** subdirectory of the LispWorks installation.

In the image shipped, ***use-motif-library*** is set to

```
(:config-file
 "use-motif-library"
 :detect-version
 (("/usr/X11R6/lib/libXm.so.2.0.1" :lesstif)
  ("/usr/X11R6/lib/libXm.so.3" :motif))
 :prefer-version
 :lesstif)
```

which causes **/usr/X11R6/lib/libXm.so.2.0.1** to be treated as Lesstif Xm 2.1 (and preferred) and **/usr/X11R6/lib/libXm.so.3** to be treated as Motif Xm 2.2. The default config-file is

<*install*>/lib/5-0-0-0/config/use-motif-library

11.3.4 Using multiple versions of Motif/Lesstif on Linux

The versions of Motif and Lesstif used by LispWorks 5.0 may not be compatible with other applications (including LispWorks 4.2). They are however compatible with LispWorks 4.4 and 4.3, so you should be able to run LispWorks 5.0, LispWorks 4.4 and LispWorks 4.3 simultaneously with either OpenMotif or Lesstif installed.

You may wish to maintain multiple versions of the Motif/Lesstif libraries in order to run various applications simultaneously. However, because the filenames of the libraries can conflict, this can only be done by installing libraries in non-standard locations.

When a library is installed in a non-standard location, you can set the environment variable **LD_LIBRARY_PATH** to allow an application to find that library.

For example, to use Motif 2.2 for LispWorks 5.0 but keep Motif 2.1.30 (which is used by LispWorks 4.2) as the default:

1. Install Motif 2.2 in some directory other than **/usr/X11R6/lib**. Let <*motiflibdir*> denote the directory containing the Motif 2.2 file **libXm.so**.
2. Set **LD_LIBRARY_PATH** to include <*motiflibdir*>.
3. In your LispWorks initialization file, add

   ```
   #+LispWorks5
   (setq x-utils::*use-motif-library*
   '(:prefer-version :motif))
   ```

   to tell LispWorks 5.0 to use Motif (otherwise it will find **libXm.so.2** and expect it to be some version of Lesstif).

Or, if you have already installed Motif 2.2 in the standard location and wish to re-install Motif 2.1 for use with LispWorks 4.2:

1. Install Motif 2.1 in some directory other than **/usr/X11R6/lib**. Let <*motiflibdir*> denote the directory containing the Motif 2.1 file **libXm.so**.
2. Set **LD_LIBRARY_PATH** to include <*motiflibdir*>.

3. In your LispWorks initialization file, add

```
#+LispWorks4.2
(setq x-utils::*use-motif-library*
'(:prefer-version 2))
```

to tell LispWorks 4.2 to use Motif.

Note: to find out which version of libXm your LispWorks 5.0 image is actually using, look in the bug form. See "Generate a bug report template" on page 107 for instructions on generating the bug form.

11.3.5 Using LispWorks for Linux on FreeBSD

LispWorks relies on FreeBSD's Linux compatibility libraries. Therefore to use external libraries with LispWorks for Linux, such as Motif/Lesstif, you will need to install Linux versions of these.

LispWorks for FreeBSD is now available, however.

11.4 Troubleshooting on FreeBSD

This section describes some of the most common problems that can occur during installation or configuration of LispWorks for FreeBSD.

See also "Troubleshooting on X11/Motif" on page 101 below for issues specific to X11/Motif.

11.4.1 Poor latency when using multiple threads

When running on FreeBSD 6.0, you may get better latency when running with threads by setting the environment variable **LIBPTHREAD_SYSTEM_SCOPE** to 1 before starting LispWorks.

11.5 Troubleshooting on UNIX

This section describes some of the most common problems that can occur during installation or configuration of LispWorks for UNIX (not including Linux).

See also “Troubleshooting on X11 / Motif” on page 101 for issues specific to X11 / Motif.

11.5.1 Problems with CD-ROM file system

Some operating systems provide tools which can mount a CD-ROM incorrectly. If your LispWorks CD-ROM appears to contain files named like this:

```
lwdoc50-unix.tar;1
```

then check the **mount** command used (“Mounting the CD-ROM” on page 79).

11.5.2 License key errors

LispWorks looks for a keyfile containing a valid license key when it is started up. If a problem occurs at this point, LispWorks exits, after first printing a keyfile error message.

There are three possible problems:

- LispWorks cannot find or read the key file.
- The key in the keyfile is incorrect.
- Your license has expired, making the key no longer valid.

An appropriate error message will appear for each case.

An unconfigured image must either be installed in the default location (library hierarchy under **/usr/lib/lispworks/lib/5-0-0-0**) or be executed in the same directory as the keyfile. If the image has been configured, check that the keyfile is in the right place and that the value of ***lispworks-directory*** is correct.

If the key is incorrect, check it against the one Lisp Support supplied. It should consist only of numerals and upper case letters (A–Z). If the key has expired, contact Lisp Support—you may be allowed to extend the key.

11.6 Troubleshooting on X11/Motif

This section describes some of the most common problems that can occur using the LispWorks X11 / Motif GUI, which is available on Linux, FreeBSD, Mac OS X and UNIX.

11.6.1 Problems with the X server

Running under X11/Motif, LispWorks may print a message saying that it is unable to connect to the X server. Check that the server is running, and that the machine the image is running on is authorized to connect to it. (See the manual entry for command **`xhost(1)`**.)

On Mac OS X, if you attempt to start the LispWorks X11/Motif GUI in Terminal.app, an error message **`Failed to open display NIL`** is printed. Instead, run LispWorks in X11.app.

11.6.2 Problems with fonts

Running under X11, LispWorks may print a message saying that it is unable to open a font and is using a default instead. The environment will still run but it may not always use the right font.

LispWorks comes configured with the fonts most commonly found with the target machine type. However the fonts supplied vary between implementations and installations. The fonts available on a particular server can be determined by using the **`xlsfonts(1)`** command. Fonts are chosen based on the X11 resources. See “X11 resources” on page 103 for more information.

It may be necessary to change the fonts used by LispWorks.

11.6.3 Problems with colors

Running under X11, on starting up the environment, or any tool within it, LispWorks may print a message saying that a particular color could not be allocated.

This problem can occur if your X color map is full. If this is the case, LispWorks cannot allocate all the colors that are specified in the X11 resources.

This may happen if you have many different colors on your screen, for instance when displaying a picture in the root window of your display.

Colors are chosen based on the X11 resources. See “X11 resources” on page 103 for more information.

To remove the problem, you can then change the resources (for example, by editing the file mentioned in “X11 resources” on page 103) to reduce the number of colors LispWorks allocates.

11.6.4 Mnemonics and Alt

Motif hardwires its mnemonic processing to use **mod1**, so we disable mnemonics if that is Lisp's **Meta** modifier to allow the Emacs-style editor to work. (The accelerator code uses the same keyboard mapping check as the mnemonics so **Alt** accelerators would also get disabled if you had them.)

11.6.5 Non-standard X11 key bindings

When using X11 / Motif, if you want Emacs-style keys **Ctrl-n, Ctrl-p** in LispWorks list panels, such as the Editor’s buffers view, add the following to the X11 resources (see Section 11.6.6):

```
!
! Enable Ctrl-n, Ctrl-p in list panels
Lispworks*XmList.translations: #override\n\
    Ctrl<Key>p : ListPrevItem()\n\
    Ctrl<Key>n : ListNextItem()
!
```

11.6.6 X11 resources

When using X11 / Motif, LispWorks reads X11 resources in the normal way, using the application class Lispworks. The file **app-defaults/Lispworks** is used to supply fallback resources. You can copy parts of this file to **~/Lispworks** or some other configuration-specific location if you wish to change these defaults, and similarly for **app-defaults/GcMonitor**.

11.6.7 Motif installation on Mac OS X

When attempting to starting the LispWorks X11 / Motif GUI when the required version of Motif is not installed, LispWorks prints the error message:

```
Error: Could not register handle for external module X-
UTILITIES::CAPIX11:
dyld: /Applications/LispWorks 5.0/lispworks-5-0-0-macos-
universal-motif can't open library:
/usr/local/lib/libXm.4.dylib (No such file or directory, errno
= 2)
.
```

Ensure you install Motif as described in Section 2.4.9.2, "The X11/Motif GUI (required on 64-bit LispWorks)". Restart X11.app and LispWorks after installation of Motif.

11.7 Updating with patches

We sometimes issue patches to the Professional and Enterprise Editions of LispWorks and LispWorks for UNIX by email or ftp.

11.7.1 Extracting simple patches

Save the email attachment to your disk.

See Section 11.7.3.2, "Private patches" below about location of your private patches.

11.7.2 If you cannot receive electronic mail

If your site has neither electronic mail nor ftp access, and you want to receive patches, you should contact Lisp Support to discuss a suitable medium for their transmission.

11.7.3 Different types of patch

There are two types of patch sent out by Lisp Support, and they have to be dealt with in different ways.

11.7.3.1 Public patches

Public patches are general patches made available to all LispWorks customers. These are typically released in bundles of multiple different patch files; each file has a number as its name. For example,

patches/system/0001/0001.nfasl (for PowerPC Mac OS X)
patches\system\0001\0001.ofasl (for x86 Windows)
patches/system/0001/0001.ufasl (for x86 Linux)
patches/system/0001/0001.ffasl (for x86 FreeBSD)
patches/system/0001/0001.64nfasl (for PowerPC64 Mac OS X)
patches/system\0001\0001.64ofasl (for x64 Windows)
patches/system/0001/0001.64ufasl (for amd64 Linux)
patches/system/0001/0001.pfasl (for HP-PA)
patches/system/0001/0001.wfasl (for SPARC)
patches/system/0001/0001.64wfasl (for SPARC 64 bit)

On receipt of a new patch bundle your system manager should update each local installation according to the installation instructions supplied with the patch bundle. This will add files to the patches subdirectory and increment the version number displayed by LispWorks.

You should consider saving a new image with the latest patches pre-loaded, as described in Section 7.4, “Saving and testing the configured image” (Mac OS X), Section 8.4, “Saving and testing the configured image” (Windows) or Section 9.4, “Saving and testing the configured image” (Linux), or Section 10.4.7, “Saving and testing the configured image” (non-Linux UNIX).

11.7.3.2 Private patches

LispWorks patches are generally released in cumulative bundles. Occasionally Lisp Support may send you individual patch binaries named e.g. **my-patch** to address a problem or implement a new feature in advance of bundled ('public') patch releases. Such patches have real names, rather than numbers, and must be loaded once they have been saved to disk. You will need to ensure that LispWorks will load your private patches on startup, after public patches have been loaded.

There is a default location for private patches, and patch loading instructions sent to you will assume this location. Therefore, on receipt of a private patch **my-patch.ufasl**, the simplest approach is to place it here. For example, on Mac OS X:

<*install*>/LispWorks 5.0/Library/lib/5-0-0-0/private-patches/my-patch.nfasl

On Windows:

<*install*>\lib\5-0-0-0\private-patches\my-patch.ofasl

On Linux:

<*install*>/lib/5-0-0-0/private-patches/my-patch.ufasl

On UNIX:

<*install*>/lib/5-0-0-0/private-patches/my-patch.pfasl (for HP-PA)
<*install*>/lib/5-0-0-0/private-patches/my-patch.wfasl (for SPARC)

You will receive a Lisp form needed to load such a patch, such as

```
(LOAD-ONE-PRIVATE-PATCH "my-patch" :SYSTEM)
```

This form should be added in the file:

private-patches/load.lisp

like the example there. **load-all-patches** loads this file, and hence all the private patches listed therein.

You may choose to save a reconfigured image with the new patch loaded - for details see the instructions in Section 7.4, “Saving and testing the configured image” (Mac OS X), Section 8.4, “Saving and testing the configured image” (Windows), Section 9.4, “Saving and testing the configured image” (Linux), or Section 10.4.7, “Saving and testing the configured image” (non-Linux UNIX). You can alternatively choose to load the patch file on startup. The option you choose will depend on how many people at your site will need access to the new patch, and how many will need access to an image without the patch loaded.

11.8 Reporting bugs

The LispWorks system is tested extensively prior to release. If you discover a new bug, in either the software or the documentation, you can submit a bug report by any of the following routes.

- email
- fax
- paper mail (post)
- telephone

The addresses are listed in Section 11.8.7. Please note that we much prefer email.

11.8.1 Check for existing fixes

Before reporting a bug, please ensure that you have the latest patches installed and loaded. Visit **www.lispworks.com/downloads/patch-selection.html** for the latest patch release.

If the bug persists, check the Lisp Knowledgebase at **www.lispworks.com/support/** for information about the problem - we may already have fixed it or found workarounds.

If you need informal advice or tips, try joining the LispWorks users' mailing list. Details are at **www.lispworks.com/support/lisp-hug.html**.

11.8.2 Generate a bug report template

Whatever method you want to use to contact us, choose **Help > Report Bug** from any tool, or use the command **Meta+X Report Bug**, or at a Lisp prompt, use **:bug-form**, for example:

```
:bug-form "foo is broken" :filename "bug-report-about-foo.txt"
```

All three methods produce a report template you can fill in. In the GUI environment we prefer you use the **Report Bug** command - do this from within the debugger if an error has been signalled.

The bug report template captures details of the Operating System and Lisp you are running, as well as a stack backtrace if your Lisp is in the debugger. There may be delays if you do not provide this essential information.

If the issue you are reporting does not signal an error, or for some other reason you are not able to supply a backtrace, we still want to see the bug report template generated from the relevant LispWorks image.

11.8.3 Add details to your bug report

Tell us if the bug is repeatable. Add instructions on how to reproduce it to the 'Description' field of the bug report form.

Include any other information you think might be relevant. This might be your code which triggers the bug. In this case, please send us a self-contained piece of code which demonstrates the problem (this is much more useful than code fragments).

Include the output of the Lisp image. In general it is not useful to edit the output, so please send it as-is. Where output files are very large and repetitive, the first and last 200 lines might be adequate.

If the problem depends on a source or resource file, please include that file with the bug report.

If the bug report falls into one of the categories below, please also include the results of a backtrace after carrying out the extra steps requested:

- If the problem seems to be compiler-related, set **`*compiler-break-on-error*`** to **`t`**, and try again.
- If the problem seems to be related to **`error`** or conditions or related functionality, trace **`error`** and **`conditions::coerce-to-condition`**, and try again.
- If the problem is in the Common LispWorks IDE, and you are receiving too many notifiers, set **`dbg::*full-windowing-debugging*`** to **`nil`** and try again. This will cause the console version of debugger to be used instead.
- If the problem occurs when compiling or loading a large system, call (**`toggle-source-debugging nil`**) and try again.
- If you ever receive any unexpected terminal output starting with the characters **`<**>`**, please send *all* of the output—however much there is of it.

  Note: terminal output is that written to **`*terminal-io*`**. Normally this is not visible when running the Mac OS X native GUI or the Windows GUI, though it is displayed in a Terminal.app or MS-DOS window if necessary.

11.8.4 Reporting crashes

Very occasionally, there are circumstances where it is not possible to generate a bug report form from the running Lisp which displays the bug. This can

occur in a delivered image which lacks the debugger, or a bug which causes lisp to crash completely. It is still useful for us to see a bug report template from your lisp image (that is, generate the template before your code is loaded or a broken call is made) so that we can at least ascertain system details.

Then, try running the lisp image with output redirected to a file. For example, make a file and **my-init-file.lisp** which loads your code that leads to the crash and run, on Mac OS X:

```
% "/Applications/LispWorks 5.0/LispWorks.app/Contents/MacOS/lispw
orks-5-0-0-macos-universal" -init my-init-file.lisp > lw.out
```

where **%** denotes a Unix shell prompt.

On Windows:

```
C:\> "Program Files\LispWorks\lispworks-5-0-0-x86-
win32.exe" -init my-init-file.lisp > lw.out
```

where **C:\>** denotes a MS-DOS prompt.

On Linux:

```
% /usr/bin/lispworks-5-0-0-x86-linux -init my-init-file.lisp >
lw.out
```

where **%** denotes a Unix shell prompt.

On UNIX (SPARC in this example):

```
% /usr/lib/lispworks/lib/5-0-0-0/config/lispworks-5-0-0-sparc-
solaris -init my-init-file.lisp > lw.out
```

11.8.5 Log Files

If your application writes a log file, add this to your report. If your application does not write a log file, consider adding it, since a log is always useful. The log should record what the program is doing, and include the output of **(room)** periodically, say every five minutes.

11.8.6 Reporting bugs in delivered images

Some delivered executables lack the debugger. It is still useful for us to see a bug report template from your Lisp image that was used to build the deliv-

ered executable. If possible, load your code and call **(require "delivery")** then generate the template.

For bugs in delivered LispWorks images, the best approach is to start with a very simple call to **deliver**, at level 0 and with the minimum of delivery keywords (**:interface :capi** and **:multiprocessing t** at most). Then deliver at increasingly severe levels. Add delivery keywords to address specific problems you find (see the *LispWorks Delivery User Guide*.for details. However, please note that you are not expected to need to add more than 6 or so delivery keywords: do contact us if you are adding more than this.)

11.8.7 Send the bug report

Email is usually the best way. Send your report to

```
lisp-support@lispworks.com
```

When we receive a bug report, we will send an automated acknowledgment, and the bug will be entered into the LispWorks bug management system. The automated reply has a subject line containing for example

```
(Lisp Support Call #12345)
```

Please be sure to include that cookie in the subject line of all subsequent messages concerning your report, to allow Lisp Support to track it.

If you cannot use email, please either:

- Fax to +44 870 2206189
- Post to Lisp Support, LispWorks Ltd, St John's Innovation Centre, Cowley Road, Cambridge, CB4 0WS, England
- Telephone: +44 1223 421860

Note: It is *very important* that you include a *stack backtrace* in your bug report wherever applicable. See “Generate a bug report template” on page 107 for details. You can always get a backtrace from within the debugger by entering **:bb** at the debugger prompt

11.8.8 Sending large files

Note: Please check with Lisp Support in advance if you are intending to send very large files via email.

11.8.9 Information for Personal Edition users

We appreciate feedback from users of LispWorks Personal Edition, and often we are able to provide advice or workarounds if you run into problems. However please bear in mind that this free product is unsupported. For informal advice and tips, try joining the LispWorks users mailing list. Details are at `www.lispworks.com/support/lisp-hug.html`.

12

Release Notes

12.1 Additional platforms supported

LispWorks now supports the following additional platforms:

- Intel-based Macintosh computers
- FreeBSD on x86
- 64-bit Linux on AMD64/EM64T systems
- 64-bit Windows x64 Editions
- 64-bit Mac OS X G5 systems
- 64-bit Solaris/SPARC

LispWorks for FreeBSD, 64-bit LispWorks for Linux, 64-bit LispWorks for Windows, 64-bit LispWorks for Macintosh and 64-bit LispWorks for Solaris are each licensed separately from each other and separately from the other platforms that LispWorks supports.

12.1.1 Nomenclature

Where we write simply "LispWorks for Linux", "LispWorks for Windows" and so on, we generally mean the 32-bit implementations. In addition, where we write "LispWorks for Macintosh" we include both PowerPC and Intel-based Macintosh computers.

The new 64-bit implementations are named explicitly "LispWorks (64-bit) for Linux" and also referred to informally as "64-bit LispWorks for Linux", and so on.

12.1.2 Universal binaries on the Macintosh

The supplied 32-bit LispWorks for Macintosh images are universal binaries which run the correct native architecture on PowerPC and Intel-based Macintosh computers by default.

A running Lisp image only supports one architecture, chosen when the image was started. On a PowerPC based Macintosh, this is always the PowerPC architecture. On an Intel-based Macintosh, it can be either the native Intel architecture or the PowerPC architecture (using Rosetta).

Functions such as **`save-image`** and **`deliver`** create an image containing only the running architecture and functions that operate on fasl files such as **`compile-file`** and **`load`** only support the running architecture.

12.1.2.1 Building a universal binary

To build a universal binary application from LispWorks for Macintosh 5.0, you will need to install LispWorks on an Intel-based Macintosh computer.

Building a new universal binary requires three steps:

1. Build the application for PowerPC.

   This can be done on your Intel machine using Rosetta

2. Build the application for Intel.

3. Combine the two applications to make a universal binary.

See **`save-universal-from-script`** for details on how to automate these steps on a single Intel-based Macintosh.

Note: You may install LispWorks on multiple machines for use at the same time only if you own multiple LispWorks licenses.

12.1.3 Running on 64-bit machines

As far as we know each of the 32-bit LispWorks implementations runs correctly in the 32-bit subsystem of the corresponding 64-bit platform.

12.1.4 LispWorks for FreeBSD documentation

Except where explicitly mentioned, information stated as specific to LispWorks for Linux applies just the same to LispWorks for FreeBSD.

12.2 Improved architecture on x86 platforms

LispWorks for Windows and LispWorks for Linux now share the architecture of the other 32-bit LispWorks implementations, resulting in the following changes:

- **`fixnum`**s now hold 30 bits of data. That is, **`most-positive-fixnum`** is 2^{29}-1.
- **`array-total-size-limit`** is now 2^{28}-1.
- **`array-rank-limit`** is now 4000.
- The Garbage Collector is improved and memory management issues specific to the LispWorks 4.4/x86 architecture are removed.

This architecture is also used on Intel-based Macintosh computers.

12.3 Native threads on Linux and FreeBSD

LispWorks for Linux and LispWorks for FreeBSD use native threads from the pthread library to implement the threads created by Lisp functions such as **`mp:process-run-function`** and **`capi:display`**.

This allows foreign code on these platforms to call blocking I/O functions without causing all Lisp threads to block.

LispWorks for Windows and LispWorks for Macintosh also use native threads as in LispWorks 4.4.

12.4 New CAPI features

See the *LispWorks CAPI Reference Manual* for more details of these.

12.4.1 Editor pane allows variable width fonts

capi:editor-pane now supports the use of variable width fonts on Windows and Motif.

This new feature supports increasing demand for a more flexible editor in end-user applications.

12.4.2 Editor cursor blinking

Editor pane cursors now blink on and off in the platform-standard ways.

New APIs allow you to control this behavior. Start with the documentation for **capi:editor-pane-blink-rate**.

12.4.3 Better drag scrolling in the editor

Scrolling of a **capi:editor-pane** now continues after the user drags the mouse from the editor pane to the area above or below the pane. The speed of scrolling depends on the distance of the mouse from the edge of the pane.

In LispWorks 4.x the pane only scrolled while the mouse was being moved.

12.4.4 Cocoa improvements

You can now create dialogs as window-modal sheets. This is supported by **capi:display-dialog**, **capi:popup-confirmer** and the various **capi:prompt-for-** functions. In LispWorks 4.4 and previous versions, all Cocoa dialogs were application-modal.

capi:popup-menu-button is a new element, useful for invoking commands.

The user can now sort a **capi:multi-column-list-panel** by clicking on the header of a column.

The OpenGL example now includes support for the **:depth-buffer** configuration option.

12.4.5 Motif libraries loaded on CAPI startup

On X11 / Motif the Motif libraries are now loaded automatically on startup of the CAPI. Therefore users with images that use CAPI on Motif but do not start the LispWorks IDE no longer need to call **x-utilities:ensure-motif-libraries**.

On Solaris, the IDE no longer prompts you to load the Motif libraries.

12.4.6 Highlight color for graphs

Graph nodes, when selected, are now highlighted using the system highlight color.

12.4.7 Custom cursors

Use the new API **capi:load-cursor** to load custom cursors from file.

Note: Some LispWorks 4.4 users had an API called **capi:create-user-cursor**. **capi:load-cursor** is the new name for **capi:create-user-cursor** and is used in just the same way.

12.4.8 New toolbar features

Toolbars have been augmented with various new features:

- A **capi:toolbar-button** can now have a dropdown menu which may be raised by various user gestures.
- For more complex user interaction, a **capi:toolbar-button** can display a **capi:interface**.
- You can specify an alternate image to be used when a **capi:toolbar-button** is selected, and there is new control over the width of text displayed on the button.
- You can control when the outlines of a **capi:toolbar-button** appear on Cocoa.

12.4.9 New text input pane features

You now have more control over the buttons associated with a **capi:text-input-pane**:

- There is a new option to control the position of the buttons, relative to the pane.
- You may add images, tooltip help and accelerators to the buttons.
- You can control the enabled state of the buttons.

Also there is a new navigation callback which runs when certain user gestures occur in the pane.

12.4.10 Text input range values can wrap

capi:text-input-range now supports wrapped values. That is, scrolling beyond the end of the range leads to the other end of the values range.

12.4.11 List panel context menu enhancements

Item data can now be passed to the context menu in a **capi:list-panel**. Also you can control the way the selection changes during the menu and whether the original selection is restored, using the new initarg **:right-click-selection-behavior**.

12.4.12 Sound API

Load, store and play sounds in your CAPI application using the new sound API. Start with the documentation for **capi:load-sound**.

12.4.13 Clipboard supports images

capi:clipboard now supports objects of type **gp:image**.

12.4.14 Window styles

A variety of CAPI window styles are now supported for relevant platforms, as documented for the **capi:interface** initarg **:window-styles**.

12.4.15 Scrolling APIs

Use **capi:get-scroll-position** to obtain the current scroll position for panes with built-in scrolling such as **capi:list-panel** and **capi:display-pane**.

capi:get-scroll-position and **capi:scroll** now work with **capi:tree-view**.

12.4.16 Input model extends support for multiple mouse button clicks

New button actions **:third-press** and **:nth-press** are available in input models for output pane mouse click gestures on Cocoa and X11 / Motif. For the details, see the documentation of *input-model* under **capi:output-pane**.

12.4.17 Create and destroy callbacks for output panes

New callbacks can be specified with the new **capi:output-pane** initargs **:create-callback** and **:destroy-callback**.

12.4.18 New push button callbacks

Two new callbacks are available for **capi:push-button**, as follows:

- The **:press-callback** runs when the user presses the button. This works on Windows and X11 / Motif.
- The **:alternate-callback**: runs when the user presses the button and there is simultaneously a modifier key held down. This works on Windows and Cocoa.

12.4.19 Enhanced metafile support

You can draw a Windows metafile to:

- An output pane, using **capi:draw-metafile**
- A Graphics Ports image, using **capi:draw-metafile-to-image**

And you can now store metafiles on the **capi:clipboard**.

12.4.20 Pinboard optimization for adding multiple objects

There is now a more efficient way for you to add multiple objects to a pinboard layout. Use the new **:add-many** keyword argument to **capi:manipulate-pinboard**.

12.4.21 Slider direction control

Control which end of a **capi:slider** has the start of the values range by using the new initarg **:start-point**.

12.4.22 Text wrapping width

capi:wrap-text-for-pane now offers control over the wrapping width, font and allows a substring to be specified.

12.4.23 Control over title gaps

capi:titled-object now allows you to specify the size of the gaps between the pane and its title and message.

12.4.24 Transparency for CAPI windows

The new **capi:interface** initarg **:transparency** controls the overall transparency of the window, on Cocoa and later Windows systems.

12.4.25 User-editable simulated listener commands

capi:interactive-pane-execute-command now supports user-editable commands.

12.4.26 Problems with buttons in a pinboard on Cocoa fixed

Problems with buttons disappearing when used within a pinboard (either directly or via **capi:contain**) have been fixed.

12.4.27 Memory leaks fixed

Various memory and resource leaks in CAPI on Windows and Cocoa are now fixed.

12.5 New graphics ports features

For details see the Graphics Ports chapters in the *LispWorks CAPI Reference Manual* and the *LispWorks CAPI User Guide*.

12.5.1 Alpha channel

Graphics Ports images now support an alpha channel on Cocoa and later versions of Windows. Start at the documentation for **`gp:load-image`** and **`gp:draw-image`**.

Color specifications include an alpha component, and the Image Access API supports manipulation of the alpha values.

12.5.2 Image Access API extended

See the documentation for **`gp:image-access-pixels-to-bgra`**.

12.5.3 System highlight colors

Graphics Ports can now use the OS highlight color, as specified via

- the Windows Control Panel (**Display > Appearance > Advanced**)
- the Mac OS X System Preferences (**General > Highlight Color**)
- the X11 / Motif resources colorHighlight and colorHighlightText

You can supply **`:background :color_highlight`** and **`:foreground :color_highlighttext`** (the latter is ignored on Mac OS X).

12.6 More new features

For details of these, see the documentation in the *LispWorks Reference Manual*, unless a manual is referenced explicitly.

12.6.1 Windows themes

The CAPI and IDE now support Microsoft Windows themes on Windows XP and Windows 2003. See the documentation for **win32:set-application-themed**.

12.6.2 X11 resources

12.6.2.1 New colors and fonts

A new set of fallback resources for application classes Lispworks and GcMonitor are supplied in the **app-defaults** directory. The new colors and fonts are based on a design by Stephane Perrot, to whom we are grateful for the contribution.

The LispWorks 4 resources are still available as an alternative in the files **Lispworks-classic** and **GcMonitor-classic**.

12.6.2.2 Control over highlight colors

The new resources colorHighlight and colorHighlightText give you control over the default highlight colors used in the various pane classes. See **app-defaults/Lispworks** for some examples.

12.6.3 Running the event loop in pure Cocoa applications

For applications which use Cocoa but not CAPI, you can now run the Cocoa event loop in the main thread. See the manual entry for **mp:initialize-multiprocessing**.

12.6.4 Programmatic dismissal of the splash screen on Windows

The function **win32:dismiss-splash-screen** makes a startup screen (as specified via the **:startup-bitmap-file** delivery keyword) disappear.

12.6.5 Preservation of user preferences

The first time you run LispWorks 5.0 it copies any user preferences stored for LispWorks 4.4.x on the installation machine, so your preferred window positions and so on are preserved. The copying is done once, and then a flag is set, so further modifications to the preferences in the old version do not affect the new version.

To add similar functionality to your LispWorks applications, call the new function **sys:copy-preferences-from-older-version**.

12.6.6 Weak arrays

There are two new APIs for weak arrays. See the entries in the *LispWorks Reference Manual* for

- **make-array** with its new **:weak** argument
- the new function **copy-to-weak-simple-vector**

12.6.7 More float types implemented on x86 platforms

LispWorks now implements 3 disjoint ANSI Common Lisp float types: **short-float**, **single-float** and **double-float**.

long-float is the same type as **double-float**.

Note: In versions 4.4 and previous of LispWorks for Windows and LispWorks for Linux all floats are **double-float**.

12.6.8 The stderr stream

You can now write to your application's stderr directly from Lisp. See **sys:make-stderr-stream**.

12.6.9 Control over symbol printing

Two new variables offer extra control over the way that symbols print:

- **lw:*print-nickname*** controls which package prefix is output, when the package name is needed.

- **sys:*print-symbols-using-bars*** controls the way that escaping is done, when it is necessary.

12.6.10 Displaying web pages

sys:open-url is now the supported way to display a page in a web browser.

Note: please convert any uses of **hqn-web:browse** or internal functions to **sys:open-url**.

12.6.11 Accessing large files

LispWorks now supports access to files larger than 4GB on Windows XP and Windows 2000, Linux, FreeBSD and Mac OS X.

64-bit LispWorks for Solaris also supports this.

12.6.12 Conditional file compilation

The function **hcl:compile-file-if-needed** is now supported. It provides a way to compile a file only if the binary file is older than the source file, or is missing.

12.6.13 Waiting on socket streams

sys:wait-for-input-streams and **sys:wait-for-input-streams-returning-first** wait for input on multiple socket streams, returning when input is available on at least one of the streams.

12.6.14 Local socket address and port

Client sockets now support local address and local port. See the documentation for **comm:open-tcp-stream** for the details.

12.6.15 Write timeouts for socket streams

Socket streams now support a write timeout. See **comm:socket-stream**.

12.6.16 file-write-date and Windows folders

file-write-date now works for directories (folders) on Windows.

12.6.17 Foreign callbacks from unknown threads

On Windows, Mac OS X, Linux and FreeBSD, foreign code can now call into Lisp in any thread, including those that LispWorks did not create.

On Windows the new function **sys:setup-for-alien-threads** sets up LispWorks to handle calls (via foreign callables) from "alien" threads, that is threads that LispWorks did not start. This setup happens automatically in LispWorks DLLs, but if there is a call into a LispWorks executable from an alien thread before **sys:setup-for-alien-threads** is called, the effect is unpredictable.

12.6.18 System-calling APIs improved

sys:call-system and **sys:call-system-showing-output** now have a more consistent syntax between Windows and all other platforms.

You can now kill subprocesses invoked by **sys:call-system-showing-output** after calling it with the new keyword argument **:kill-process-on-abort**.

12.6.19 Stack extension

hcl:extend-current-stack and **hcl:current-stack-length** are now implemented on Windows and Linux.

Automatic stack extension is now implemented for all platforms. See the manual page for **sys:*stack-overflow-behaviour*** in the *LispWorks Reference Manual* for details of how to control this.

12.6.20 Regular expression symbol search

The new function **lw:regexp-find-symbols** returns a list of symbols with names that match a supplied regular expression. This is rather like **cl:apropos-list**, but more powerful.

12.6.21 Regular expression end-of-line matching

lw:find-regexp-in-string and other regular expression APIs now allow $ to match end-of-line from within () or |.

12.7 IDE changes

See the *Common LispWorks User Guide* for details of the features mentioned.

12.7.1 New Application Builder tool

The Application Builder allows standalone applications to be generated and tested from within the IDE, in the Professional and Enterprise Editions.

When first invoked, the tool is configured to build the "Hello World" delivery example. Choose **Build > Build** to build this and **Build > Run** to run it.

12.7.2 New Symbol Browser tool

The Symbol Browser tool gives you the functionality of **cl:apropos** and more, integrated with the IDE. Enter a regular expression into the Regexp pane and press **Return** to see all symbols matching that pattern.

More details of each symbol can be obtained by clicking on the symbol in the middle pane. Various filtering options are available to control which symbols are displayed.

12.7.3 New Tracer tool

The Tracer provided control over which functions are traced. It shows the tracing options for the selected function and allows the options to be modified by double-clicking on ther name of the function

The tool also collects the output from tracing. The **Output Data** tab shows the arguments and values in a form that can be used to launch other tools such as the Inspector. The **Output Text** tab shows the output in textual format.

12.7.4 Window-modal preferences dialogs on Cocoa

Most of the dialogs such as those raised by the **File > Print...** and **Preferences...** commands are now window-modal on Cocoa.

12.7.5 Editor tool improved font support

The LispWorks Editor tool now allows the use of variable width fonts, including CJK fonts such as those with Japanese kanji characters.

Note: The LispWorks Editor was designed as a code editor which is why we previously supported only fixed width fonts, and of course these are still appropriate for code editing.

12.7.6 Process Browser supports auto-update

You can now make the Process Browser tool update automatically at a predetermined frequency by setting the Update Frequency in its Preferences dialog.

12.7.7 Bug report uses Editor tool on Unix platforms

LispWorks for Unix no longer uses sendmail and the Mailer tool for bug reports. The command **Help > Report Bug** now creates an Editor tool showing the bug report template on all platforms.

12.8 Editor changes

See the *LispWorks Editor User Guide* for details of these changes.

12.8.1 Selection of comment lines

The editor now treats a comment line as a form when selecting and copying with the mouse.

12.8.2 Apropos invokes the Symbol Browser

The command **Apropos** (**Ctrl+H A**) has been changed so that it raises a Symbol Browser tool with the supplied string. If a non-**nil** prefix argument is supplied, then any special characters in the string are escaped so that the regular expression match does not treat them specially.

The new command **Apropos Command** offers the previous functionality of **Apropos**: it displays a list of editor commands, variables, and attributes whose names contain the string, in a Help window.

12.8.3 Regular expression end-of-line matching

Regular expression searching by the commands **Regexp Search** and friends now allows $ to match end-of-line from within () or |.

12.9 Foreign Language interface changes

See the *LispWorks Foreign Language Interface User Guide and Reference Manual* for details of these changes.

12.9.1 Default language

The default language in **fli:define-foreign-function** and the other FLI definers is now **:ansi-c**.

12.9.2 :long-long type

The FLI now supports the type **:long-long** on some platforms.

12.10 COM/Automation changes

This section applies only to Windows platforms. See the *LispWorks COM/Automation User Guide and Reference Manual* for details.

12.10.1 Registry utilties

There are three new utilities for querying the registry.

See the documentation for **com:find-clsid**, **com:find-component-value** and **com:find-component-tlb**.

12.10.2 Loading the type library for a component

The new keyword option **:component-name** for the **:midl-type-library-file defsystem** member type allows you to specify the type library associated with the component rather than naming the type library file directly.

12.11 Common SQL changes

12.11.1 Native support for MySQL

There is a new native interface to MySQL. See the section "Using MySQL" in the *LispWorks User Guide*.

12.11.2 Oracle OCI interface

There is now a native interface that uses the Oracle Calls Interface (OCI).

Use **:database-type :oracle** to connect to any supported version of Oracle.

For backwards compatibility, you may also use **:database-type :oracle8** to connect to Oracle 8.

For Oracle versions 9i (release 2) or 10g, Common SQL now uses the OCI.

12.11.3 Oracle LOBs

Oracle large object types such as BLOB and CLOB can now be created and accessed with Common SQL. See the "Oracle LOB interface" section in the *LispWorks User Guide*.

12.11.4 Oracle native support on Mac OS X

The native Oracle interface is now supported on Mac OS X running on PowerPC. It will be supported on Intel-based Macintoshes when the client libraries become available from Oracle.

12.11.5 Naming connections

sql:connect takes various new options and has a new mechanism for naming conections. See the *LispWorks Reference Manual* for details.

12.11.6 Common SQL operators for calling database functions

sql-operator and **sql-boolean-operator** are new.

The pseudo operator **function**, introduced in LispWorks 4.2, has been renamed **sql-function** to avoid potential name clashes with database identifiers.

See the *LispWorks User Guide* for more information about these new operators.

12.11.7 New query macro

sql:simple-do-query is an easy-to-use variant of **sql:do-query**.

12.11.8 def-view-class :writer slot option

The **:writer** slot option in the **sql:def-view-class** macro now names the writer function using the specified name, in the same way as **defclass** would. In previous releases, **:writer foo** would generate a writer called **(setf foo)**.

12.12 KnowledgeWorks changes

12.12.1 Inferencing with multiple concurrent threads

KnowledgeWorks has been extended to support inferencing from multiple concurrent threads. See the section "Inferencing states" in the *KnowledgeWorks and Prolog User Guide*.

12.12.2 Truth maintenance

A rule can now include a clause that specifies logical dependencies amongst KnowledgeWorks objects. These dependencies are preserved in any objects that are created directly as a result of the rule firing. See the section "Forward Chaining Syntax" in the *KnowledgeWorks and Prolog User Guide*, specifically the documentation for the **`kw:logical`** clause in **`kw:defrel`**.

12.12.3 TEST clause optimization

KnowledgeWorks **`kw:test`** clauses are now evaluated less frequently, in particular they are no longer evaluated when a surrounding forward-condition clause changes from matching an object in the object-base to not matching any objects.

12.12.4 Prolog optimization

Prolog is now compiled more efficiently when debugging information is not required. This allows for faster execution and reduces memory consumption.

12.12.5 Detection of unbound variables in NOT clauses

`kw:defrule` for forward rules now detects cases where a NOT clause refers to an unbound variable.

12.12.6 kw:no-debug controls compile-time optimization

In LispWorks 5.0 **`(kw:no-debug)`** improves the compilation of both forward and backward chaining rules. In LispWorks 4.4 and previous versions, it only optimized some aspects of forward chaining at runtime.

12.13 Application delivery changes

See the *LispWorks Delivery User Guide* and the *Common LispWorks User Guide* for details of these changes.

12.13.1 Post delivery function

A new delivery keyword **:post-delivery-function** allows you to specify a function to be called after delivery. **:post-delivery-function** supersedes **:exit-after-delivery** which is no longer available.

12.13.2 Changes to some deliver keywords

On the Windows and Linux platforms there are changes to some of the **deliver** keyword arguments, including:

```
:diagnostics-file
:symbol-names-action
:keep-complex-numbers
:keep-trans-numbers
:numeric
```

The following **deliver** keywords are no longer needed and are not implemented:

```
:symbols-to-precache-symbol-names
:keep-lexer
:keep-bignum-numbers
:keep-float-numbers
:keep-ratio-numbers
```

12.13.3 New Application Builder tool

There is now an Application Builder in the Common LispWorks IDE, which is a convenient way to invoke Delivery.

12.14 CLOS/MOP changes

12.14.1 Changes for better MOP compliance

The **:default-initargs** in a **defclass** form are passed as **:direct-default-initargs** to the corresponding call of **clos:ensure-class** (and passed on to **clos:ensure-class-using-class** and **initialize-instance**). This is as specified in the MOP. In LispWorks 4.4 and previous versions, these were passed as **:default-initargs**.

effective-slot-definition-class and **initialize-instance** on **effective-slot-definition** now receive an **:allocation** initarg as specified in the MOP. Note, however, that there is no mechanism to store arbitrary **:allocation** values.

(setf generic-function-name) and **(setf class-name)** are now implemented using **reinitialize-instance**.

The **generic-function declare** option is now initialized by the initarg **:declarations**. In LispWorks 4.4 and previous versions, this initarg was **:declare**, contrary to the MOP.

add-method now calls **remove-method**.

The **reinitialize-instance** protocol for class metaobjects has been corrected, now calling **add-direct-subclass**, **remove-direct-subclass** and **class-direct-subclass** as specified in the MOP.

Initialization and reinitialization of class metaobjects now provides the correct default for **:direct-superclasses** in the **shared-initialize** method. This allows a new metaclass to provide its own default superclass instead of **standard-object** or **funcallable-standard-object**. In LispWorks 4.4 and previous, **:direct-superclasses** is added at **defclass** macroexpansion time, which has the effect that for a new metaclass which provides its own default superclass one first has to remove **standard-object** or **funcallable-standard-object** from the direct superclasses. This is contrary to the MOP.

12.15 Other changes

12.15.1 Runtime library requirement on Windows

LispWorks for Windows now requires the Microsoft Visual Studio runtime library **msvcr80.dll**. The LispWorks installer installs this DLL if it is not present.

Applications you build with LispWorks for Windows also require this DLL, so you must ensure it is available on target machines.

12.15.2 Changes in cl:*features*

The feature **:win32** is deprecated. Use **:mswindows** instead.

For a full description including information about the features used to distinguish new LispWorks implementations and platforms, see the entry for ***features*** in the *LispWorks Reference Manual*.

12.15.3 Using 32-bit and 64-bit foreign libraries

Load only libraries of the correct architecture into LispWorks. You will need to obtain (or build) a 32-bit library for use with 32-bit LispWorks and / or a 64-bit library for use with 64-bit LispWorks. You may conditionalize the argument to **fli:register-module** with the feature **:lispworks-64bit**.

Note: On Linux, you may see a spurious "No such file or directory" error when loading a foreign library of the wrogn architecture. The spurious message might be localized.

12.15.4 Foreign libraries loader API changed on UNIX platforms

LispWorks for UNIX now uses **fli:register-module** to load foreign code. This is the API which is already available on other platforms. Please replace your uses of **link-load:read-foreign-modules**.

12.15.5 Change to implicitly-created foreign modules on Windows and Linux

If **:module** is passed to **fli:define-foreign-function** or **fli:make-pointer** and the module has not been registered, then the automatically created module now has *connection-style* **:manual**. (In LispWorks 4.x, the *connection-style* here is **:automatic)**.

This change prevents these implicit module definitions from being used accidentally to look up symbols that did not specify a module.

12.15.6 Foreign modules named by symbols on Windows and Linux

Modules named by symbols must be registered with a **:real-name** to provide the filename.

In LispWorks 4.x, the name of the symbol is used, but this can lead to unexpected loading of shared libraries.

12.15.7 Name conversion by the FLI

The *foreign-name* in **fli:define-foreign-callable** and the value of the **:symbol-name** argument to **fli:make-pointer** are now converted in the same way as the *foreign-symbol* or *foreign-string* in **fli:define-foreign-function**.

12.15.8 Larger heaps in 32-bit LispWorks for Linux

The Lisp heap in 32-bit LispWorks for Linux can now reach around 2.4GB, depending on the configuration of the machine. See the section "Layout of Memory" in the *LispWorks User Guide* for details.

LispWorks for Linux 4.4 is effectively limited to around 1.4GB on some Linux systems.

12.15.9 call-arguments-limit raised

The Common Lisp constant **call-arguments-limit** is now 2047, on all LispWorks platforms.

In LispWorks 4.4 the value was 255 on Windows and Linux, and 300 on the Macintosh and other Unix platforms.

Note: It is usually bad programming style to pass such a large number of arguments, though long argument lists can occur in automatically-generated code.

12.15.10 Allocating static objects

The recommended way to allocate an array in the static area is now to call **make-array** with **:allocation :static**. See the documentation under **make-array** in the *LispWorks Reference Manual*.

sys:in-static-area and **sys:switch-static-allocation** are deprecated.

12.15.11 file-namestring portability change

file-namestring now returns **nil** for directory pathnames. This follows the practice of other Common Lisp implementations by returning **nil** for pathnames with unspecified *name* and *type* components.

In LispWorks 4.4 and previous versions, the value was an empty string.

12.15.12 defgeneric :method-combination changed

The arguments in a **defgeneric :method-combination** clause are no longer evaluated. In LispWorks 4.4 and previous versions, these arguments were evaluated, but the ANSI Common Lisp standard does not specify this.

12.15.13 clos:slot-fixed-allocation

clos:slot-fixed-allocation has been removed, because it is no longer implemented.

12.15.14 Compilation and evaluation environments

The compilation and evaluation environments in LispWorks 5.0 (on all platforms) now behave exactly as in LispWorks 4.4.x for Macintosh and LispWorks 4.4.x for Unix.

12.15.15 Non-compliant compile time side-effects removed

The following Common Lisp functions and macros no longer have compile time side-effects:

make-package
defpackage
in-package
shadow
shadowing-import
export
unexport
use-package
unuse-package
import

In LispWorks 4.4 and previous versions there were side-effects, which was non-compliant with the ANSI Common Lisp standard.

12.15.16 Multiprocessing API altered

process-event-queue is no longer exported in the multiprocessing package. The accessor to get a process's mailbox (event queue) is **mp:process-mailbox**.

12.15.17 copy-tree improved

On Windows and Linux, **copy-tree** is no longer recursive and hence can handle deeper trees than in versions 4.4 and previous.

12.15.18 New fasl filename extensions

The default fasl filename extension is now **ofasl** in LispWorks for Windows and **ufasl** in LispWorks for Linux.

The default fasl filename extension is **ffasl** in LispWorks for FreeBSD, **64ofasl** in 64-bit LispWorks for Windows, **64ufasl** in 64-bit LispWorks for Linux, **64nfasl** in 64-bit LispWorks for Macintosh and **64wfasl** in 64-bit LispWorks for Solaris.

When running 32-bit LispWorks for Macintosh natively on an Intel Macintosh the default fasl filename extension is **xfasl**. However when running 32-bit LispWorks on a PowerPC Macintosh, or under Rosetta on an Intel Macintosh, the default fasl filename extension is **nfasl**, as in LispWorks 4.4.

Fasls (binary files) generated by **compile-file** in previous versions of LispWorks cannot be loaded into LispWorks 5.0 in any case, but if you have code that assumes the LispWorks 4.4 extensions (**fsl**, **ufsl** respectively) then conditionalize it, something like this:

```
#+LispWorks4 "fsl" #-LispWorks4 "ofasl"
```

Fasl file types for the various LispWorks platforms are documented under **compile-file** in the *LispWorks Reference Manual*.

12.15.19 Loading old data files

Binary files created with **hcl:dump-forms-to-file** or **hcl:with-output-to-fasl-file** in LispWorks 4.4 or LispWorks 4.3 can be loaded into LispWorks 5.0 using **sys:load-data-file**.

Note: because the default fasl extensions have changed on some platforms (see "New fasl filename extensions" above), you may need to do something like this example in LispWorks 5.0 for Windows:

```
(let ((sys:*binary-file-type* "fsl"))
  (sys:load-data-file "C:/lww44.fsl"))
```

12.15.20 Loading logical pathnames

When **load** is called with a logical pathname the system now checks the translated pathname in case its **pathname-type** is different. That is, given

```
(translate-logical-pathname "HOST:FOO;bar.BIN")
->
#p"W:\\Development\\foo\\bar.nfasl"
```

then **load** looks for the truename. In LispWorks 4.4 and previous versions, **load** can incorrectly load it as a text file because **.BIN** was not recognised as a binary file type.

12.15.21 time macro report format

All three of User, System and Elapsed times are now reported in a consistent format. This is:

- ***secs*.*micros*** if less than 60 seconds
- ***hours*:*minutes*:*secs*.*micros*** if 60 seconds or more.

For example:

```
User time    =        2.310
System time  =        0.000
Elapsed time =  0:01:02.409
```

12.15.22 Definition protection removed for keywords

In the default configuration LispWorks no longer warns when you give a function or macro definition to a keyword symbol. This behavior is controlled by the variable **`*packages-for-warn-on-redefinition*`**.

Such definitions do not cause any problems for LispWorks itself, though there are potential clashes between user level modules and it is not recommended that you do this.

12.15.23 Reader handles Unicode quote characters

The standard readtable now handles the Unicode quote (**`#\U+2019`**) and backquote (**`#\U+2018`**) characters like ASCII quote and backquote. Also the Unicode Double Comma (**`#\U+201C`**) and Turned Double Comma (**`#\U+201D`**) characters (in that order) delimit a string.

12.15.24 Soft Hyphen not a graphic character

The character at Unicode code point 173 (Soft Hyphen) is no longer graphic in the sense of **`graphic-char-p`** (because Unicode now marks it as a control character).

12.15.25 Licensing problems with Macintosh hostnames fixed

32-bit LispWorks for Macintosh (on PowerPC and Intel-based machines) no longer uses the machine's Unix hostname when storing the license. This change fixes license validation failures seen when you disconnect or connect the machine and the Internet after you install LispWorks.

The recommended workaround of setting **`HOSTNAME=-AUTOMATIC-`** is no longer necessary for users of 32-bit LispWorks for Macintosh. However, 64-bit LispWorks for Macintosh users should read “Failure to start when disconnected from the Internet” on page 95.

12.16 Documentation changes

12.16.1 HyperSpec search

Use of the Common Lisp HyperSpec via the LispWorks **Help > Search...** command is improved, by using the extensive glossary as part of the index which is searched.

12.16.2 PostScript format manuals

The LispWorks distribution no longer contains documentation in the PostScript format. However this is still available at **`www.lispworks.com/documentation`**.

12.17 Known Problems

12.17.1 Problems with LispWorks for Macintosh

The Motif GUI doesn't work "out of the box" with Fink because LispWorks does not look for **`libXm`** etc in **`/sw/lib/`**.

Functions run by **`mp:process-run-function`** have their standard streams connected to **`*terminal-io*`** (which is not normally visible). Possibly when the IDE is running, output should be connected to the Background Output buffer.

Reading from **`*terminal-io*`**, closing Terminal.app and then reading again gets end of file.

12.17.2 Problems with the LispWorks IDE on Cocoa

Multithreading in the CAPI is different from other platforms. In particular, all windows run in a single thread, whereas on other platforms there is a thread per window.

The debugger currently doesn't work for errors in Cocoa Event Loop or Editor Command Loop threads. However, there is a **Get Backtrace** button so you can obtain a backtrace and also a **Debug Snapshot** button which aborts from the error but displays a debugger with a copy (snapshot) of the stack where the error occurred.

The online documentation interface currently starts a new browser window each time.

Setting **`*enter-debugger-directly*`** to **`t`** can allow the undebuggable processes to enter the debugger, resulting in the UI freezing.

Inspecting a long list (for example, 1000 items) via the Listener's **`Inspect Star`** editor command prompts you about truncation in a random window. If you cancel, the inspect is still displayed.

The editor's Help about help (**`Control+h Control+h`**) dialog is messy because it assumed that a fixed width font is being used.

The Services menu is not functional.

It is impossible to interrupt loops in the Cocoa Event Loop process.

The **Definitions > Compile** and **Definitions > Evaluate** menu options cause multiple "Press space to continue" messages to be displayed and happen interleaved rather than sequentially.

The **Buffers > Compile** and **Buffers > Evaluate** menu options cause multiple "Press space to continue" messages to be displayed and happen interleaved rather than sequentially.

12.17.3 Problems with CAPI and Graphics Ports on Cocoa

Some graphics state parameters are ignored, in particular operation, stipple, pattern, fill-style and mask (other than a rectangle).

LispWorks ignores the System Preferences setting for the smallest font size to smooth.

There is no support for state images or checkboxes in **`capi:tree-view`**.

`capi:with-page` does not work, because Cocoa tries to control page printing.

The **:help-callback** initarg is only implemented for the **:tooltip** value of the type argument.

The **:visible-border** initarg only works for scrolling panes.

Programmatic scrolling of **capi:list-panel** etc is not implemented.

Caret movement and selection setting in **capi:text-input-pane** is implemented, but note that it works only for the focussed pane.

capi:docking-layout doesn't support (un)docking.

There is no meta key in the input-model of **capi:output-pane**. Note that, in the editor when using Emacs emulation, the **Escape** key can be used as a prefix.

There has been no testing with 256 color displays.

There is no visual feedback for dead-key processing, for example **Option+n** is the tidle dead-key.

The graph pane's plan mode rectangle doesn't redraw when moved or resized.

Some pinboard code uses **:operation boole-xor** which is not implemented.

All menu items are disabled when a dialog is on the screen. However, the **Command+X**, **Command+C** and **Command+V** shortcuts work within text panes

There is no way to make the close icon on a window show the "modified" state (**NSWindow:setDocumentEdited**).

capi:editor-pane will only work with fonts whose widths are (almost) integral for example Monaco 10, 15, 20 pt etc but not Monaco 12 pt. The nearest good size is used instead.

The default menu bar is visible when the current window has no menu bar.

capi:tree-view is slow for a large number (thousands) of items.

The editor displays decomposed characters as separate glyphs.

The **:gap** option is not supported for the columns of **capi:multi-column-list-panel**.

capi:display-dialog ignores the specified **:x** and **:y** coords of the dialog (for drop-down sheets the coords are not relevant and for dialogs which are separate windows Cocoa forces the window to be in the top-center of the screen).

12.18 Binary Incompatibilty

If you have binaries (fasl files) which were compiled using LispWorks 4.4.5 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 5.0