LispWorks®

Release Notes and Installation Guide

Version 4.4.6

Copyright and Trademarks

LispWorks *Release Notes and Installation Guide*

Version 4.4.6

October 2005

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

tel: +44 1223 421860
fax: +44 870 2206189
toll-free number for calls from the US:
tel: 877 759 8839

www.lispworks.com

Contents

1 Introduction

1.1 LispWorks Editions

LispWorks is available in four product editions on the Macintosh, Windows and Linux platforms:

- Personal Edition - includes a native graphical IDE on all platforms, 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 non-Linux UNIX platforms (includes Solaris, HP-UX, Tru64 UNIX) LispWorks is licensed differently, as detailed in "LispWorks for UNIX" on page 3.

1.1.1 Current Version

The current version of LispWorks Personal Edition is 4.4.6.

A patch release upgrading the commercial product to version 4.4.6 will be available in October 2005.

1.1.2 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 heap limit has been raised in version 4.4.6, allowing more programs to be run.
- 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.

1.1.3 Professional Edition

LispWorks 4.4.5 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

1.1.4 Enterprise Edition

LispWorks 4.4.5 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

1.1.5 Academic Edition

LispWorks 4.4.5 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 non-Linux UNIX platforms (includes Solaris, HP-UX, Tru64 UNIX) the Edition model described above does not apply.

LispWorks for UNIX 4.4.5 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 details of LispWorks products visit the LispWorks Web Site:

`www.lispworks.com`

To purchase LispWorks please use the contact details at:

`www.lispworks.com/buy`

1.4 About this Guide

This document is an installation guide and release notes for LispWorks 4.4.5 on Mac OS X, Windows, Linux 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 5, “Installation on UNIX”.

The following four chapters explain in detail everything necessary to configure, run, and test LispWorks 4.4.5. Choose the chapter for your platform: Chapter 6, “Configuration Details on Mac OS X”, Chapter 7, “Configuration Details on Windows”, Chapter 8, “Configuration Details on Linux” or Chapter 9, “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 10, “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 11, “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 for Macintosh 4.4.6. Chapter 6 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 GUIs. 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. However, if you later decide to install the other option, you can simply run the installer again.

LispWorks for Macintosh Personal Edition supports only the native Mac OS X 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 copy of Adobe® Reader®. This can be downloaded from the Adobe website at **`www.adobe.com`**.

2.3 Software and hardware requirements

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

Hardware Requirements	Software Requirements
32MB of memory, preferably 64MB	Mac OS X version 10.3.x or 10.4.x
137MB of disk space for Enterprise Edition plus documentation	OpenMotif 2.1 (or higher) if you want to run the X11/Motif GUI.

Table 2.1

2.4 Installing LispWorks for Macintosh

2.4.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as an installer containing version 4.4.5 and a downloadable patch bundle which upgrades LispWorks to version 4.4.6. You need to complete the main installation before adding patches.

LispWorks Academic and Personal Editions are supplied as installers containing version 4.4.6.

2.4.2 Information for Beta testers

Users of LispWorks 4.4 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 4.4.5. You can run the Beta installer and select the **Uninstall** option (and then remove any patches) or simply drag the LispWorks 4.4 folder to the trash.

2.4.3 Information for LispWorks 4.4 users

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

You can install LispWorks Personal Edition 4.4.6 in the same location as LispWorks Personal Edition 4.4.5. A new **`LispWorks Personal 4.4.6`** folder will be created.

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

You should have a **`LispWorksInstaller`** application, visible with the ringed "LW" icon in the Mac OS X Finder.

If you have downloaded LispWorks, you may need to extract the installer from a file **`LispWorks4.4.5.dmg`** - simply double-click to extract it.

If you have received LispWorks on a CD-ROM, the **`LispWorksInstaller`** application should be visible in the MacOS folder after mounting the CD-ROM.

Double-click on **`LispWorksInstaller`** to launch the installer.

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

Choose where the `LispWorks 4.4.5` folder, or the `LispWorks Personal 4.4.6` folder, will be created and click **Continue**.

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

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**.

2.4.9.2 The X11/Motif GUI

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 2.1, available at **`www.motifzone.org`**.

LispWorks requires Motif version 2.1. The currently available OpenMotif runtime is Motif version 2.1.30 even though the downloaded file is called openmotif-2.2.0-OSX.pkg.

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 the LispWorks Personal Edition software: then the software and the documentation will reside in the same **`LispWorks Personal 4.4.6`** folder. However, the content of the Personal Edition documentation is unchanged since version 4.4.5.

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 previously installed Xanalys LispWorks 4.4 then you should use your existing license data. This should be on a label inside the Xanalys shipment folder. Contact **`lisp-keys@lispworks.com`** if you cannot find it.

If yours is a new LispWorks license then your key is supplied on a label on the folder containing the CD-ROM, or will be supplied to you via email from Lisp Support.

2.4.12 Add LispWorks to the Dock

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**.

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 4.4.5` or `LispWorks Personal 4.4.6` 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.

2.5 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.

If you added LispWorks to the Dock during installation, you can also start LispWorks from this. 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-4-4-5-darwin` image.

See Section 6.3, “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-4-4-6-darwin`, 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-4-4-6-darwin`.

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 4.4.5"
xterm% ./lispworks-4-4-5-darwin-motif
LispWorks(R): The Common Lisp Programming Environment
Copyright (C) 1987-2005 LispWorks Ltd.  All rights reserved.
Version 4.4.5
Saved by LispWorks as lispworks-4-4-5-darwin-motif, at 18 Feb
2005 16:08
User dubya on octane
; Loading text file /Applications/LispWorks 4.4.5/Library/lib/4-
4-0-0/config/siteinit.lisp
;  Loading text file /Applications/LispWorks 4.4.5/Library/lib/4-
4-0-0/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 4.4.5"
xterm% ./lispworks-4-4-5-darwin-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-4-4-5-darwin-motif`** image.

See Section 6.3, “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-4-4-6-darwin-motif`**.

3 Installation on Windows

This chapter is an installation guide for LispWorks for Windows 4.4.6. 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.

3.1 Documentation

The LispWorks documentation set is available in two electronic forms: HTML and PDF. You can either access the documentation directly from the CD-ROM or from a disk drive if you chose to install it as specified in Section 3.2, “Installing LispWorks for Windows”.

Note: if you have installed LispWorks for Windows from a downloaded self-extracting executable, then you do not have the option to access the documentation from a CD-ROM.

The HTML documentation can be used from within the Common LispWorks environment via the **Help** menu. It is also available from the **Start** menu under **Start > Programs > LispWorks 4.4 > Browsable 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 > Programs > LispWorks 4.4 > Printable Documentation**. To view and print these files, you will need a copy of 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 and Enterprise Editions are supplied as an installer containing version 4.4.5 and a downloadable patch bundle which upgrades LispWorks to version 4.4.6. You need to complete the main installation before adding patches.

LispWorks Academic and Personal Editions are supplied as installers containing version 4.4.6.

3.2.2 Installing over previous versions: Professional and Enterprise Editions

You should not install LispWorks 4.4.5 in the same location as a previous version of LispWorks. If you always select the default installation location, then this situation will not occur, since LispWorks now installs by default in **`Program Files\LispWorks`** rather than **`Program Files\Xanalys\LispWorks`**.

You can install LispWorks 4.4.5 without uninstalling previous versions of LispWorks such as Xanalys LispWorks 4.4 or Xanalys LispWorks 4.3.

3.2.3 Installing over previous versions: Personal Edition

We recommend that you uninstall LispWorks Personal Edition 4.4.5 before installing LispWorks Personal Edition 4.4.6. You may install version 4.4.6 in the same location as version 4.4.5, but be aware that then they will share the same library, and each uninstaller will remove that library.

If you already have the documentation for LispWorks Personal Edition 4.4.5 installed, then you do not need to install the LispWorks Personal Edition documentation again since it is unchanged in version 4.4.6. Simply install the LispWorks Personal Edition 4.4.6 software in the same location as the documentation (after uninstalling LispWorks Personal Edition 4.4.5).

3.2.4 Information for Beta testers

Users of LispWorks 4.4 Beta should completely uninstall it before installing LispWorks 4.4.5. Remember to remove any patches added since the initial beta release.

3.2.5 To install the Professional or Enterprise Edition

The LispWorks for Windows CD-ROM has an autostart feature. After you insert the CD-ROM in the drive, a dialog appears anouncing the start of the installation process. (You can also run **`Windows\Setup.exe`** on the CD-ROM to start the installer.) Follow the instructions on screen.

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 previously installed Xanalys LispWorks 4.4 then the serial number should appear in the installer's dialog box. Add your existing license key which should be on a label inside the Xanalys shipment folder. Contact **`lisp-keys@lispworks.com`** if you cannot find it.

If yours is a new LispWorks license then your key is supplied on a label on the folder containing the CD-ROM, or will be supplied to you via email from Lisp Support.

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 when the LispWorks CD-ROM is in the CD drive.

You will also need to decide whether to install the Printable Documentation. If you choose not to install this, you will be able to access the Printable (PDF) Documentation only when the LispWorks CD-ROM is in the CD-ROM drive.

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 > Programs > LispWorks 4.4 > 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 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 > Programs > LispWorks Academic 4.4 > LispWorks**.

4

Installation on Linux

This chapter is an installation guide for LispWorks for Linux 4.4.6. 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.

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 6-9) or SuSE or Mandrake with kernel version 2.2+
152MB of disk space for Enterprise Edition plus documentation	OpenMotif 2.2 (or higher) or Lesstif 0.93.18 (or higher)

Table 4.1

Hardware Requirements	Software Requirements
	Netscape Web browser for viewing on-line documentation

Table 4.1

4.1.1 Motif libraries

LispWorks 4.4 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) are installed on your machine. The LispWorks CD-ROM includes Lesstif 0.93.95b as distributed from **`www.lesstif.org`**. If you need to install it first, contact your systems administrator for help.

If you are not installing from the CD-ROM, see **`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 4.4.x and LispWorks 4.3 simultaneously with either OpenMotif or Lesstif installed.

4.1.2 Disk requirements during installation

LispWorks requires about 43MB to install without documentation. Installing the documentation adds about 109MB (Professional / Enterprise / Academic) or 45MB (Personal) to this. A full installation of the Enterprise Edition with all documentation and loadable modules requires about 152MB.

The Professional / Enterprise / Academic documentation includes two printable formats: PDF (23MB) and PostScript (41MB). You may delete any of these that you do not need. They are available at **`www.lispworks.com/reference`** in any case.

4.1.3 Memory requirements at runtime

LispWorks 4.4 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.

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.

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 locate the LispWorks for Linux files in the `Linux` 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 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 lwlpro-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 for Linux 4.4 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 **`Linux`** directory 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 and Enterprise Editions are supplied as an installer containing version 4.4.5 and a downloadable patch bundle which upgrades LispWorks to version 4.4.6. You need to complete the main installation before adding patches.

LispWorks Academic and Personal Editions are supplied as installers containing version 4.4.6.

4.4.2 Information for Beta testers

Users of LispWorks 4.4 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 4.4.5.

See “Uninstalling LispWorks for Linux” on page 30 for instructions.

4.4.3 Installation from the binary RPM file

We recommend that you use RPM 2.5.1 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 4.4 Beta installed, please uninstall it before installing this product. See Section 4.9, “Uninstalling LispWorks for Linux”.

If you already have Xanalys L:ispWorks 4.4 installed, you can simply upgrade it to LispWorks 4.4.5 as described below.

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. By default LispWorks is installed in **`/usr/lib/LispWorks`** and a symbolic link to the executable is placed in **`/usr/bin/lispworks-4450`**. 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 84 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.

Note: The default RPM installation location is different to LispWorks 4.1 (which used **`/usr/local/lib`**).

Note: For the Personal Edition, use **`lispworks-personal-4.4-*.i386.rpm`** wherever **`lispworks-4.4-*.i386.rpm`** is mentioned in this document. See Section 1.1.2, “Personal Edition” for more information specific to the Personal Edition. Similarly, for the Academic Edition use **`lispworks-academic-4.4-*.i386.rpm.`**

4.4.3.1 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-4.4-`*`n`*`.i386.rpm`**.
   n denotes a build number. For the LispWorks 4.4.5 release, *n* = 3.
2. Install or upgrade LispWorks in the standard RPM way, for example:

   ```
   rpm --install lispworks-4.4-n.i386.rpm
   ```

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

   ```
   rpm --install --prefix <directory> lispworks-4.4-n.i386.rpm
   ```

   installs LispWorks in <*directory*>.

   If you have LispWorks Personal Edition 4.4.5 installed, and do not wish to uninstall it, then you need to upgrade (rather than install) to LispWorks Personal Edition 4.4.6:

   ```
   rpm -Uh lispworks-personal-4.4-2.i386.rpm
   ```

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.

4.4.3.2 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-4.4-`*`n`*`.i386.rpm`	CLIM 2.0	Professional
`lispworks-kw-4.4-`*`n`*`.i386.rpm`	KnowledgeWorks	Enterprise
`lispworks-corba-4.4-`*`n`*`.i386.rpm`	LispWorks ORB	Enterprise
`lispworks-sql-4.4-`*`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 4.4.3.1).

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

4.4.3.3 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-4.4-`*`n`*`.i386.rpm`	CLIM 2.0	Academic
`lispworks-academic-kw-4.4-`*`n`*`.i386.rpm`	KnowledgeWorks	Academic

Table 4.3 File distributions for layered products in the Academic Edition

File Distribution	Layered Product	LispWorks Edition
`lispworks-academic-corba-4.4-n.i386.rpm`	LispWorks ORB	Academic
`lispworks-academic-sql-4.4-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 4.4.3.2.

4.4.3.4 Documentation and saving space

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

Documentation is installed by default in the **`lib/4-4-0-0/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-4.4-n.i386.rpm
```

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

```
rpm --install --prefix <directory> --replacepkgs
lispworks-4.4-n.i386.rpm
```

4.4.3.5 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 Professional and Enterprise Editions are:

`lwlproent445.tar.gz`	LispWorks Professional and Enterprise image, modules and examples
`lwldoc445.tar.gz`	Documentation in HTML, PDF and Postscript formats

The gzipped files for the Personal Edition are:

`lwlper446.tar.gz`	LispWorks Personal image, modules and examples
`lwlperdoc445.tar.gz`	Documentation in HTML format (unchanged since version 4.4.5)

The gzipped files for the Academic Edition are:

`lwlac446.tar.gz`	LispWorks Academic image, modules and examples
`lwldoc446.tar.gz`	Documentation in HTML, PDF and Postscript 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 **`lwlpro-install.sh`** (or **`lwlper-install.sh`** for the Personal Edition, **`lwlac-install.sh`** for the Academic Edition)).

For example, to install the Personal Edition and documentation in the default location (**`/usr/local/lib/LispWorksPersonal`**) from the CD-ROM mounted on **`/mnt/cdrom`** you would use:

```
cd /mnt/cdrom
sh lwlper-install.sh
```

Or, to install the Professional Edition in **`/usr/lispworks`**, without documentation, from a CD-ROM mounted on **`/mnt/cdrom1`** you would use:

```
cd /mnt/cdrom1
sh lwlpro-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 8.2, “License keys”.

4.6 Running LispWorks

The LispWorks executable is located in **`/usr/lib/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-4460`	Personal Edition
`lispworks-4450`	Professional or Enterprise Edition
`lispworks-academic-4460`	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 81 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:

```
lispworks-4450 --lwlicenseserial SERIALNUMBER --lwlicensekey KEY
```

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

If you have previously installed Xanalys LispWorks 4.4 then you should use your existing license data. This should be on a label inside the Xanalys shipment folder. Contact **`lisp-keys@lispworks.com`** if you cannot find it.

If yours is a new LispWorks license then your license data is supplied on a label on the folder containing the CD-ROM, or will be supplied to you via email from Lisp Support.

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 8, “Configuration Details on Linux”.

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/4-4-0-0/manual/offline`** directory contains PostScript language versions of the manuals, as well as PDF versions.

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

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-4.4
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named **`lispworks-patches4.4`**. 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 UNIX

5.1 Introduction

This chapter is a brief installation guide for UNIX LispWorks 4.4.5. Chapter 9 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 9, "Configuration Details on UNIX".

5.2 Extracting software from the CD-ROM

LispWorks 4.4.5 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.

5.2.1 Finding out which CD-ROM files you need

The following table shows products found on the CD-ROM:

Product	Product code
LispWorks	**`lispworks`**
CLIM 2.0	**`clim-lispworks`**
KnowledgeWorks	**`kw-lispworks`**
LispWorks ORB	**`corba-lispworks`**

Table 5.1 Products and associated codes

The following table shows the platforms upon which these products are supported:

Platform	Hardware code	OS code
Compaq Tru64 (OSF1 4.0F & later)	**`alpha`**	**`alpha`**
HP PA (HP-UX 11x)	**`hp-pa`**	**`hp-pa11`**
Sun Sparc (Solaris 2.6 & later)	**`sparc`**	**`sparc-solaris`**

Table 5.2 Platforms and associated codes

To work out which files you need, remember that:

- You have to install LispWorks in order to use *any* of the products.
- Files are named <*hardware code*>**`-`**<*product code*>**`445.tar`**.

See the tables above for the *hardware code* and *product code.*

LispWorks Developer comes in two archives. The image, libraries and examples are found in the platform-specific file, and documentation is found in a common file. The documentation file is **`doc-lispworks445.tar`**. This contains documentation for Common Lisp and LispWorks and the layered products KnowledgeWorks, LispWorks ORB and CLIM.

Each layered product is contained in an additional archive.

If, for example, you wanted to unpack KnowledgeWorks 4.4.5 for a Sun Solaris machine, you would need the file **`sparc-kw-lispworks445.tar`**.

5.2.2 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 and decide which **`tar`** files you need. These will include <*hardware code*>**`-lispworks445.tar`** and **`doc-lispworks445.tar`**.
4. Use the following command to unpack each **`tar`** file:

   `% tar -xof` ***filename***

The LispWorks image file can be found in the resulting **`lib/4-4-0-0/config`** directory, named according to the operating system, platform, and LispWorks version number, as follows:

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

Thus, an image named **`lispworks-4-4-5-hp-pa11`** would be a LispWorks 4.4.5 image for use on an HP PA machine running HP-UX 11.

5.3 Installing the image and runtime files

Now put the image file from the **`config`** directory into a suitable location on your filesystem.

5. Move everything you unpacked to an appropriate place on your filesystem. For example:

   ```
   % mkdir -p /usr/lib/lispworks/lib
   % mv lib/4-4-0-0 /usr/lib/lispworks/lib
   ```

   The prefix **`/usr/lib/lispworks`** can be changed if `required`.

The lispworks image must be able to find its library. The library location is contained in the Lisp variable **`*lispworks-directory*`**. The value of this variable in the supplied image is the pathname **`#P"/usr/lib/lispworks/"`**. So, if you put the library in **`/usr/lib/lispworks`**, then **`*lispworks-directory*`** should not be changed.

6. If you put the library elsewhere, then the simplest approach is to make a symbolic link **`/usr/lib/lispworks/lib`**, Alternatively, you can set **`*lispworks-directory*`** to the correct value when configuring your local lispworks image (see Section 5.5 below).

5.4 Keyfiles and License server permissions

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.

5.4.1 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 4.1/4.2 also work for LispWorks 4.4.5. However, keys issued for LispWorks 3 do not.

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 10.7, “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 10.7.7, “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/4-4-0-0/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 5.3 above)
- **`lib/4-4-0-0/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 76.

5.4.2 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*.

5.5 Configuring the LispWorks image

Now you can configure the LispWorks image to your taste. In the distribution directory **`config`** (where you found the LispWorks image file) 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.

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.

5.5.1 Saving a configured image

Make a copy of **`config/configure.lisp`** called **`config/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 something like:

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

2. Change directory to the **`config`** subdirectory of the LispWorks installation directory, for example:

   ```
   % cd /usr/lib/lispworks/lib/4-4-0-0/config
   ```

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

   ```
   % lispworks-4-4-5-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 “Keyfiles and License server permissions” on page 34

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.

5.5.2 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 > (disassemble 'car)
   ```

   The disassembler is a load-on-demand feature, so if the installation is correct you will see messages reporting that the disassembler 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"`**.

5.6 Using the Documentation

Documentation in HTML, PostScript and PDF formats is provided in **`doc-lispworks44.tar`** on the CD-ROM. If you want to access the documentation, you must unpack this archive.

HTML documentation is installed in the **`lib/4-4-0-0/manual/online`** subdirectory of the LispWorks library, and can be accessed via the **`Help`** menu in the Common LispWorks IDE.

The printable formats are installed in the **`lib/4-4-0-0/manual/offline`** subdirectory of the LispWorks library.

5.7 Using Layered Products

To use each of LispWorks ORB, CLIM 2.0 and KnowledgeWorks, you must

1. Unpack the appropriate archive. See “Finding out which CD-ROM files you need” on page 32

2. Obtain the required key and put in your keyfile. See “Keyfiles and License server permissions” on page 34.
3. 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 36

6

Configuration Details on Mac OS X

6.1 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"

6.2 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/4-4-0-0/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.

6.3 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.

6.3.1 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.

6.3.2 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.

6.3.3 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 6.4, below, and Section 6.5, "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 6.4, below, and Section 6.5, "Initializing LispWorks" for further details.

6.4 Saving and testing the configured image

Make a copy of **config/configure.lisp** called **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 something like:

```
(load-all-patches)
(load "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 4.4.5/My LispWorks.app")
            #-:cocoa
            "my-lispworks-motif")
(quit)
```

   Note 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**

2. Change directory to the directory containing the LispWorks image to configure. For the native Mac OS X LispWorks image:

   ```
   % cd "/Applications/LispWorks 4.4.5/LispWorks.app/Contents/MacOS"
   ```

   or for the X11/Motif LispWorks image:

   ```
   % cd "/Applications/LispWorks 4.4.5"
   ```

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-4-4-5-darwin -siteinit - -init /tmp/save-config.lisp
   ```

   or

   ```
   % lispworks-4-4-5-darwin-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.

6.4.1 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 > (disassemble 'car)
   ```

   Before the machine code of the function **`car`** is printed, the system should load the disassembler from the **`load-on-demand`** Library directory.

6.4.2 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 42 but pass the **`:environment`** argument to **`save-image`**. For example:

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

6.5 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 4.4.5/LispWorks.app/Contents/MacOS/lis
pworks-4-4-5-darwin" -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 4.4.5/LispWorks.app/Contents/MacOS/lis
pworks-4-4-5-darwin" -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.

6.6 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")`.

6.7 The Common SQL interface

LispWorks for Macintosh supports Common SQL through the ODBC interface.

6.7.1 Loading Common SQL

To load Common SQL enter:

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

At runtime call:

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

and then you can connect to any installed ODBC datasource.

See the *LispWorks User Guide* for further information.

6.7.2 Supported databases

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

6.7.3 Special considerations when using Common SQL

6.7.3.1 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.

6.7.3.2 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.

6.7.3.3 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.

6.8 Common Prolog and KnowledgeWorks

As in LispWorks 4.3, Common Prolog is bundled with KnowledgeWorks rather than with LispWorks. KnowledgeWorks is loaded by using:

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

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

7
Configuration Details on Windows

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 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”
- “Loading the Common SQL interface”
- “Common Prolog and KnowledgeWorks”

7.2 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.

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, 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.

7.3.2 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 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 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 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 **`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 `C:\temp\save-config.lisp`, containing something like:

   ```
   (load-all-patches)
   (load "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-4450 -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-4460.exe
```

7.4.1 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 > (disassemble 'car)
   ```

   Before the machine code of the function **car** is printed, the system should load the disassembler from the **modules** 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 51 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. 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:

```
C:\Program Files\LispWorks>lispworks-4450 -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:

```
C:\Program Files\LispWorks>lispworks-4450 -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 LispWorks 4.4.5 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)
```

7.6.1 Running the CLIM demos

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

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

This creates a new window, containing a menu listing all the demos. Choose the demo you wish to see. The CLIM demos are quick sketches of possible applications which demonstrate a variety of CLIM programming techniques. They are not robust, production-quality applications with complete error checking, but they can provide you with some ideas.

You can also call demos individually with the following functions:

Bicycle gearing	`(clim-user::do-bicycle-gearing)`
Custom output records	`(clim-user::do-scigraph)`
Peek	`(clim-user::do-peek)`
Browser	`(clim-browser::do-browser)`
Ico demo	`(clim-demo::do-ico)`
Bitmap editor	`(clim-demo::do-bitmap-editor)`
Graphics editor	`(clim-graphics-editor::do-graphics-editor)`
Color chooser	`(clim-demo::do-color-chooser)`
Plotting demo	`(clim-demo::do-plot-demo)`
Thinkadot	`(clim-demo::do-thinkadot)`
Address book	`(clim-demo::do-address-book)`
15 puzzle	`(clim-demo::do-puzzle)`
Flight planner	`(clim-demo::do-flight-planner)`
CAD demo	`(clim-demo::do-cad-demo)`
Graphics demos	`(clim-demo::do-graphics-demo)`
Lisp listener	`(clim-demo::do-lisp-listener)`
Test suite	`(clim-test::do-test-suite)`

The sources for all the demos are included. The test suite is a collection of examples of CLIM's capabilities. The testsuite examples are simple and succinct, so we recommend examining their sources for examples of CLIM's functionality that you may want to employ.

7.7 Loading the Common SQL interface

The Common SQL interface requires ODBC. To load the Common SQL interface to use ODBC enter:

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

At runtime call:

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

and then you can connect to any installed ODBC datasource.

See the *LispWorks User Guide* for further information.

7.8 Common Prolog and KnowledgeWorks

As in LispWorks 4.3, Common Prolog is bundled with KnowledgeWorks rather than with LispWorks. KnowledgeWorks is loaded by using:

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

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

8 Configuration Details on Linux

8.1 Introduction

This chapter explains how to get your LispWorks Professional or Enterprise 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 4, Installation on Linux.

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 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 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/4-4-0-0/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.

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`** 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.

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 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 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 **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 something like:

   ```
   (load-all-patches)
   (load "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-4450 -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.

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. 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 > (disassemble 'car)
   ```

   Before the machine code of the function **`car`** is printed, the system should load the disassembler from the **`modules`** 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 60 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. ~ 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-4450 -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-4450 -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 4.4.5 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 creates a new window, containing a menu listing all the demos. Choose the demo you wish to see. The CLIM demos are quick sketches of possible applications which demonstrate a variety of CLIM programming techniques. They are not robust, production-quality applications with complete error checking, but they can provide you with some ideas.

You can also call demos individually with the following functions:

Bicycle gearing	`(clim-user::do-bicycle-gearing)`
Custom output records	`(clim-user::do-scigraph)`
Peek	`(clim-user::do-peek)`
Browser	`(clim-browser::do-browser)`
Ico demo	`(clim-demo::do-ico)`
Bitmap editor	`(clim-demo::do-bitmap-editor)`
Graphics editor	`(clim-graphics-editor::do-graphics-editor)`
Color chooser	`(clim-demo::do-color-chooser)`
Plotting demo	`(clim-demo::do-plot-demo)`
Thinkadot	`(clim-demo::do-thinkadot)`
Address book	`(clim-demo::do-address-book)`
15 puzzle	`(clim-demo::do-puzzle)`
Flight planner	`(clim-demo::do-flight-planner)`
CAD demo	`(clim-demo::do-cad-demo)`
Graphics demos	`(clim-demo::do-graphics-demo)`
Lisp listener	`(clim-demo::do-lisp-listener)`
Test suite	`(clim-test::do-test-suite)`

The sources for all the demos are included. The test suite is a collection of examples of CLIM's capabilities. The testsuite examples are simple and succinct, so we recommend examining their sources for examples of CLIM's functionality that you may want to employ.

8.7 Loading the Common SQL interface

The Common SQL interface requires Oracle version 8 or ODBC. To load the Common SQL interface to use ODBC enter:

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

At runtime call:

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

and then you can connect to any installed ODBC datasource.

See the *LispWorks User Guide* for further information.

8.8 Common Prolog and KnowledgeWorks

As in LispWorks 4.3, Common Prolog is bundled with KnowledgeWorks rather than with LispWorks. KnowledgeWorks is loaded by using:

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

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

9

Configuration Details on UNIX

9.1 Memory Requirements

This section discusses LispWorks 4.4.5 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.

9.1.1 Disk requirements

The basic LispWorks software requires up to 53MB of disk workspace during the installation process, depending on the platform.

Once configured, basic disk requirements are around 30 MB (again, depending upon the platform) for the image and associated libraries.

Installing the documentation adds 96 MB to this. You can delete some of these files if you wish, for example you might not need the PDF manuals in **`lib/4-4-0-0/manual/offline/pdf`** (20Mb) and/or the PostScript manuals in **`lib/4-4-0-0/manual/offline/ps`** (44Mb). However, note that the **Help** menu commands will not work if you corrupt the **`lib/4-4-0-0/manual/online`** directory of the LispWorks library.

Installing layered products adds approximately 9Mb (CLIM 2.0), 2Mb (KnowledgeWorks) and 3.5Mb (LispWorks ORB)

9.1.2 Memory requirements at runtime

LispWorks 4.4.5 requires an absolute minimum of 32 MB of user RAM in order to run. More minimizes swapping and therefore improves performance. With 64 MB RAM, LispWorks will perform well.

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 20 MB of swap space for the image alone, and whatever else is necessary to accommodate your application. LispWorks will run with less than 20 MB of swap space, but performance will suffer.

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.

9.2 Software Requirements

The LispWorks for UNIX GUI requires X11 release 5 or above, and Motif version 1.2.4 on Solaris and Tru64 UNIX, or Motif version 2.1 on HP 11.

9.3 Unpacking the CD-ROM

This section explains the organization of the LispWorks 4.4.5 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.

9.3.1 The LispWorks 4.4.5 CD-ROM

The CD-ROM contains images for LispWorks 4.4.5 and associated products on your platform or platforms.

9.3.1.1 CD-ROM format

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

9.3.1.2 Products on the CD-ROM

There are images for the following products on the LispWorks 4.4.5 CD-ROM:

- LispWorks 4.4.5
- CLIM 2.0
- KnowledgeWorks
- LispWorks ORB

The Transducer, which was provided as a separate image in LispWorks 3.2, is now supplied as a module called **`delivery`**, which is installed as part of LispWorks. It may be loaded by **`(require "delivery")`**.

9.3.1.3 Image names

The supplied LispWorks image, as found in the **`config`** directory, 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-4-4-5-sparc-solaris`** is the such as LispWorks 4.4.5 image for use on Sun Sparc Solaris machines.

9.3.2 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.

9.3.3 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.

9.3.3.1 Compaq Tru64 UNIX (DEC Alpha, OSF-1)

To mount:

```
mount -t cdfs -o noversion /dev/disk/cdrom0a /mount-point
```

where *mount-point* is the directory over which you wish to mount the CD-ROM. The device designation **/dev/disk/cdrom0a** may vary.

To unmount:

```
umount /dev/disk/cdrom0a
```

Again, use the appropriate device designation for your hardware.

9.3.3.2 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.

9.3.3.3 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_4_4/
```

To unmount:

```
umount /cdrom/lw_4_4/
```

9.3.4 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.

9.3.4.1 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.

9.3.4.2 Keeping your old LispWorks installation

You can install LispWorks 4.4.5 in the same directory as previous versions of LispWorks such as LispWorks 4.3. This is because all the 4.4 files are stored in a subdirectory called **`lib/4-4-0-0`**.

You must recompile all your code with the LispWorks 4.4 compiler.

Programs compiled in previous versions of LispWorks such as LispWorks 4.3 do not run in a LispWorks 4.4.5 image.

9.3.4.3 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 KnowledgeWorks on the Tru64/Alpha, with the CD-ROM mounted at **`/cdrom`**, you would type

```
tar -xof /cdrom/alpha-kw-lispworks44.tar
```

9.3.4.4 Compaq Tru64 UNIX (DEC Alpha, OSF-1)

The files you need to unpack for the various products are shown below.

Product	Files
LispWorks	**`alpha-lispworks44.tar`** **`doc-lispworks44.tar`**
CLIM 2.0	**`alpha-clim-lispworks44.tar`**
KnowledgeWorks	**`alpha-kw-lispworks44.tar`**
LispWorks ORB	**`alpha-corba-lispworks44.tar`**

Table 9.1 Products and associated files for Tru64 UNIX

The LispWorks image is

```
./lib/4-4-0-0/config/lispworks-4-4-5-alpha
```

9.3.4.5 HP UX (HP Precision Architecture)

The files you need to unpack for the various HP products are shown below.

Product	Files
LispWorks	**`hp-pa-lispworks44.tar`** **`doc-lispworks44.tar`**
CLIM 2.0	**`hp-pa-clim-lispworks44.tar`**
KnowledgeWorks	**`hp-pa-kw-lispworks44.tar`**
LispWorks ORB	**`hp-pa-corba-lispworks44.tar`**

Table 9.2 Products and associated filenames for HP PA

The LispWorks image for HP11 is

```
./lib/4-4-0-0/config/lispworks-4-4-5-hp-pa11
```

9.3.4.6 Solaris (Sun Sparc)

The files you need to unpack for the various Sun Solaris products are shown below.

Product	Files
LispWorks	**`sparc-lispworks44.tar`** **`doc-lispworks44.tar`**
CLIM 2.0	**`sparc-clim-lispworks44.tar`**
Knowledgeworks	**`sparc-kw-lispworks44.tar`**
LispWorks ORB	**`sparc-corba-lispworks44.tar`**

Table 9.3 Products and associated filenames for Sun Solaris

The LispWorks image is

```
./lib/4-4-0-0/config/lispworks-4-4-5-sparc-solaris
```

9.4 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 9.3, “Unpacking the CD-ROM”.

9.4.1 Introduction

We assume that the X Window System and Motif (version 1.2.4, or version 2.1 for HP11) 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

9.4.2 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.

As noted in Section 9.3.4.1 on page 69, once installed, the basic 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 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/4-4-0-0/`**. It also contains a subdirectory called **`hqn_ls`**.

Among the directories on this subdirectory are the following:

- **`config`** — the generic executable LispWorks image (see Section 9.3 for details of image names) and various files that can be adjusted in order to customize the image (see Section 9.4.6 on page 76).
- **`app-defaults`** — X/Motif resources for LispWorks and the GC Monitor.
- **`postscript`** — printer descriptions for the CAPI printing interface.
- **`etc`** — the executable for the GC 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 **`doc-lispworks44.tar`**:

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

By default, all these directories are assumed to reside beneath **`/usr/lib/lispworks/lib/4-4-0-0/`**, although you may place the **`lib`** directory somewhere else.

9.4.3 Printing copies of the LispWorks documentation

LispWorks documentation is no longer supplied in printed form. If you are a paid-up LispWorks customer, you may print extra copies of the manuals found in the LispWorks distribution, provided that each copy includes the complete copyright notice.

The **`offline`** directory contains PostScript and PDF versions, of each manual.

9.4.4 Keyfiles and how to obtain them

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.

9.4.4.1 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/4-4-0-0/config`** (see "If you are using the keyfile system" on page 34.
- **`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/4-4-0-0/config`**.

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

9.4.4.2 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 4), 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.

9.4.4.3 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 93.

Once you have the key, write it into a file in one of the places listed in Section 9.4.4.1, and start up the LispWorks image.

9.4.5 The License Server

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.

9.4.6 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.)

9.4.6.1 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.

9.4.6.2 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 9.4.7 below, and see also Section 9.5, "Initializing LispWorks" for further details.

9.4.7 Saving and testing the configured image

Make a copy of **`config/configure.lisp`** called **`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 configuration directory, for example:

   ```
   unix% cd /usr/lib/lispworks/lib/4-4-0-0/config
   ```

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

```
unix% lispworks-4-4-5-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 74.

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

   ```
   CL-USER 1 > (load "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.

9.4.7.1 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 **config**.
2. Start the new image up.
3. Test the load-on-demand system. Type:

   ```
   CL-USER 1 > (disassemble ’car)
   ```

   Before the machine code of the function **car** is printed, the system should load the disassembler 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.

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. 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.

10

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.

10.1 Troubleshooting

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

10.1.1 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 Mac OS X and Linux, this is also a possible cause of the problem:

- The machine name has changed since LispWorks was installed.

On Mac OS X and Linux, the key is expected to be stored in a keyfile, and an appropriate error message is printed for each case. On Mac OS X, you will need to look in the console for this error message.

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

10.1.2 Failure of the load-on-demand system

Module files are in the modules directory **`lib/4-4-0-0/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.

10.2 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 10.5, "Troubleshooting on X11/Motif" below for issues specific to X11/Motif.

10.2.1 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 **`LispWorksInstaller`** 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.

10.2.2 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 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-4-4-5-darwin --lwlicenseserial **** --lwlicensekey
****
```

Then the LispWorks license check will always lookup the expected hostname.

10.2.3 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**.

10.3 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 88 below for issues specific to X11 / Motif.

10.3.1 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-4.4-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-4.4-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-4.4-1
#
```

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-4.4-1.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-4.4-1.i386.rpm
```

10.3.2 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/4-4-0-0/config/use-motif-library`**

10.3.3 Using multiple versions of Motif/Lesstif on Linux

The versions of Motif and Lesstif used by LispWorks 4.4 may not be compatible with other applications (including LispWorks 4.2). They are however compatible with LispWorks 4.3, so you should be able to run LispWorks 4.4.5, 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 4.4.5 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

   ```
   #+LispWorks4.4
   (setq x-utils::*use-motif-library*
   '(:prefer-version :motif))
   ```

   to tell LispWorks 4.4.5 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 4.4.5 image is actually using, look in the bug form. See “Generate a bug report template” on page 94 for instructions on generating the bug form.

10.3.4 Using LispWorks on FreeBSD

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

10.4 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 88 for issues specific to X11/Motif.

10.4.1 Problems with CD-ROM file system

Some operating systems provide tools which can mount a CD-ROM incorrectly. If your LispWorks distribution appears to consist of files named e.g.

```
doc-lispworks44.tar;1
```

then check the `mount` command used (“Mounting the CD-ROM” on page 68).

10.4.2 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/4-4-0-0`**) 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.

10.5 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, Mac OS X and UNIX.

10.5.1 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.

10.5.2 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 89 for more information.

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

10.5.3 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 89 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 89) to reduce the number of colors LispWorks allocates.

10.5.4 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.)

10.5.5 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 10.5.6):

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

10.5.6 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, and you can copy parts of this file to **`~/Lispworks`** or some other configuration-specific location if you wish to change these defaults.

10.5.7 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 4.4.5/lispworks-4-4-5-darwin-
motif can't open library: /usr/local/lib/libXm.2.dylib (No
such file or directory, errno = 2)
.
```

Ensure you install Motif as described in Section 2.4.9.2, "The X11/Motif GUI". Restart X11.app and LispWorks after installation of Motif.

10.5.8 Loading Motif

On some Unix platforms (currently Solaris2), the Motif libraries have not been preloaded into LispWorks (this is to handle differences between OS versions). When you start the environment, you are prompted to load the libraries. An image can be built with the libraries preloaded by calling **`x-utilities:ensure-motif-libraries`** and resaving the image.

Please note that **`ensure-motif-libraries`** is no longer in the **`capi`** package.

When you deliver an X Windows application on Solaris (including CAPI applications) you should put this call in your delivery script:

```
(in-package "CL-USER")
(load-all-patches)
(load "my-application")
(x-utilities:ensure-motif-libraries)
(deliver ’my-startup-function "my-app" 5 :interface :capi)
```

10.6 Updating with patches

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

10.6.1 Extracting simple patches

Save the email attachment to your disk.

See Section 10.6.3.2, “Private patches” below about location of your private patches.

10.6.2 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.

10.6.3 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.

10.6.3.1 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, on Mac OS X:

```
patches/system/0001/0001.nfasl
```

On Windows:

```
patches\system\0001\0001.fsl
```

On Linux:

```
patches/system/0001/0001.ufsl
```

On UNIX:

`patches/system/0001/0001.pfasl` (for HP-PA)
`patches/system/0001/0001.wfasl` (for SPARC)
`patches/system/0001/0001.afasl` (for ALPHA)

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 6.4, “Saving and testing the configured image” (Mac OS X), Section 7.4, “Saving and testing the configured image” (Windows) or Section 8.4, “Saving and testing the configured image” (Linux), or Section 9.4.7, “Saving and testing the configured image” (non-Linux UNIX).

10.6.3.2 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.ufsl`**, the simplest approach is to place it here. For example, on Mac OS X:

```
<install>/LispWorks 4.4.5/Library/lib/4-4-0-0/private-
patches/my-patch.nfasl
```

On Windows:

```
<install>\lib\4-4-0-0\private-patches\my-patch.fsl
```

On Linux:

```
<install>/lib/4-4-0-0/private-patches/my-patch.ufsl
```

On UNIX:

```
<install>/lib/4-4-0-0/private-patches/my-patch.pfasl (for HP-PA)
<install>/lib/4-4-0-0/private-patches/my-patch.wfasl (for SPARC)
<install>/lib/4-4-0-0/private-patches/my-patch.afasl (for ALPHA)
```

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 6.4, “Saving and testing the configured image” (Mac OS X), Section 7.4, “Saving and testing the configured image” (Windows), Section 8.4, “Saving and testing the configured image” (Linux), or Section 9.4.7, “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.

10.7 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.

- electronic mail (email) via the Internet
- facsimile (fax) machine
- paper mail (post)
- telephone

The addresses are listed in Section 10.7.7. Please note that we much prefer email.

10.7.1 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/community/lisp-hug.html`**.

10.7.2 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.

10.7.3 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.

10.7.4 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
4.4.5/LispWorks.app/Contents/MacOS/lispworks-4-4-5-darwin" -init
my-init-file.lisp > lw.out
```

where **`%`** denotes a Unix shell prompt.

On Windows:

```
C:\> "Program Files\LispWorks\lispworks-4450.exe" -init my-init-
file.lisp > lw.out
```

where **`C:\>`** denotes a MS-DOS prompt.

On Linux:

```
% /usr/bin/lispworks-4450 -init my-init-file.lisp > lw.out
```

where **`%`** denotes a Unix shell prompt.

On UNIX (SPARC in this example):

```
% /usr/lib/lispworks/lib/4-4-0-0/config/lispworks-4-4-5-sparc-
solaris -init my-init-file.lisp > lw.out
```

10.7.5 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.

10.7.6 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 delivered 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.)

10.7.7 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 94 for details. You can always get a backtrace from within the debugger by entering **`:bb`** at the debugger prompt

10.7.8 Sending large files

Note: Please check with Lisp Support in advance if you are intending to send very large files via email.

10.7.9 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/community/lisp-hug.html`**.

11 Release Notes

11.1 Changes in LispWorks 4.4.6

This section documents changes, primarily bug fixes, in LispWorks 4.4.6 relative to LispWorks 4.4.5.

11.1.1 gethash return value

A bug with the return value of read-modify-write **`gethash`** operations is fixed.

11.1.2 error compiling abs

An ’illegal type specifier’ error when compiling **`abs`** with an argument of type **`complex`** no longer occurs.

11.1.3 Better bounds checking

`replace` (and hence **`(setf subseq)`**) now do better checks of the sequence bounds.

`adjust-array` now signals error for an out of range dimensions argument.

11.1.4 GC improvements

The garbage collector has been improved on Windows and Linux (for example, during **`compile-file`**).

Improved performance of the garbage collector when doing many allocations in the static area.

11.1.5 Stack overflow during compilation

A stack overflow that occured while compiling is now prevented.

11.1.6 Large image clean-down fix

A bug in **`clean-down`** on large images on Windows and Linux is now fixed.

11.1.7 copy-tree improved

`copy-tree` now performs better on Windows and Linux.

11.1.8 Fix for bignum times

`encode-universal-time` and **`decode-universal-time`** now work with times whose number of days since 1900 is a bignum.

11.1.9 Pretty printing on a null broadcast-stream

Errors when pretty-printing to a null **`broadcast-stream`** are now prevented.

11.1.10 Safer timers

A timer with a broken function is now less destructive.

11.1.11 Stack overflow problems

Some rare stack overflows on Windows and Darwin no longer occur.

11.1.12 Problem after failure to open a file

On Unix a bug whereby LispWorks uses all the available CPU after failing to open a file due to permission problems is fixed.

11.1.13 Walking lambda forms

A bug in the walker's handling of **`lambda`** forms is fixed.

11.1.14 Fixes in the Editor

A problem when printing an editor stream object is fixed.

The first scroll operation in a window now works correctly in MS Windows and Mac OS Editor emulation.

11.1.15 Fix in class redefinition

An error from **`(setf accessor-method-slot-definition)`** when redefining a class whose slot accessor has been defined again with **`defmethod`** is fixed.

11.1.16 Stepper tool fix

A bug in the Stepper tool has been fixed.

11.1.17 Fixes for CAPI on Cocoa

The menu bar menus update now correctly on Mac OS X 10.4 (Tiger).

Images with transparent backgrounds now work on Mac OS X 10.4 (Tiger).

The **`capi:slider`** callback can now receive the **`:drag`** operation when the user clicks and drags.

A problem with the menu switching back to the default application menu when switching back to LispWorks when a dialog is active is fixed.

Printing now obeys the scale set by the user in the Page Setup dialog for page on demand printing and in the printer metrics. Text printing no longer produces duplicate pages.

Invoking menu commands from the application interface now produces native dialogs.

The **Format for** dropdown on **`capi:page-setup-dialog`** now works on Mac OS 10.4 (Tiger).

Preview printing is fixed on Mac OS X 10.4 (Tiger).

A **`mp:process-send`** error when showing a Carbon window from the Listener is fixed.

Scaled printing with a mask has been fixed by making it work in user coordinates, and the jobname option now works for CAPI printing.

There is no longer unnecessary resizing when setting a **`capi:layout-description`** to include **`:divider`**.

The font height is now calculated correctly for **`capi:multi-line-text-input-pane`**.

11.1.18 Fixes for CAPI on Windows

The margins are now unaltered after setting the font in a **`capi:text-input-pane`** or a **`capi:display-pane`**.

There is now less redrawing when switching layouts.

Setting the background of **`capi:layout`** and **`capi:simple-pane`** now causes the window and its (possibly transparent) children to be redrawn.

Toolbars now draw correctly on a layout that has a background color, including **`capi:pinboard-layout`** and when using **`capi:contain`**.

Keyboard input is case-insensitive again for selection in **`capi:option-pane`**.

`capi:title-pane` is now transparent by default, so it works with Windows XP themes.

11.1.19 Fixes for CAPI

There is no longer unnecessary resizing when setting a **`capi:layout-description`** to include another layout.

11.1.20 Fixes for COM/Automation

Some methods with optional OUT parameters have been corrected. This fixes some errors of the form:

```
"The subscript 120 exceeds the limit 99 for the first dimension of
the array #(#S(COM::VARTYPE-CONVERTER ..."
```

`(:SAFEARRAY x)` is now treated as **`(:POINTER SAFEARRAY)`** to allow more uses in type libraries.

A bug seen when calling methods defined with **`PROPPUTREF`** is fixed.

11.1.21 Fixes in Delivery

Analysis now works when there are compiler macros.

The **`:call-count`** and **`:clos-info`** options now work for CAPI/Cocoa applications.

11.1.22 Fixes for CLIM

Redisplay of overlapping streams is fixed.

Scrolling when part of the window is hidden is fixed, on Windows.

11.1.23 Fixes in Common SQL

'Unresolved external' errors when loading the Oracle 10g shared library on HP-UX are fixed.

Incorrect SQL parenthesis balancing when using **`sql:insert-records`** with **`:query`** is now fixed.

`operator` can now be used as a field name again.

11.1.24 Larger heap in LispWorks Personal Edition

The heap limit in the Personal Edition version 4.4.6 has been relaxed, such that it can grow by more than twice the growth allowed in version 4.4.5. Therefore more programs can now be run in LispWorks Personal Edition.

11.2 Changes in LispWorks 4.4.5

This section documents changes, primarily bug fixes, in LispWorks 4.4.5 relative to LispWorks 4.4 as released by Xanalys. The subsequent sections describe the new features and known problems in LispWorks 4.4; they apply to LispWorks 4.4.5 as well.

11.2.1 Fixes for Mozilla

A bug that prevented the second use of Firefox and Mozilla browsers from the LispWorks **Help** menu on X11 / Motif has been fixed.

A bug that hindered use of the Mozilla browser from the LispWorks **Help** menu on Windows has also been fixed.

11.2.2 Fix for bus error with wide characters

After calling **`(set-default-character-element-type 'simple-char)`** and saving or delivering a LispWorks for Macintosh image, a bus error was signalled on start-up of the new image. This bug is now fixed.

11.2.3 Fix for MOP accessor method bug

A bug whereby the Metaobject Protocol function **`clos:accessor-method-slot-definition`** could return an obselete slot definition object is now fixed.

11.2.4 Spurious warning after structure redefinition

Redefinition of a structure class no longer signals a spurious warning.

11.2.5 String inequality

A bug causing **`string/=`** to return the value **`t`** rather than the numerical *mismatch-index* in some circumstances is now fixed.

11.2.6 Fix for bug with character eql specializers

A delivered application containing methods specializing on **`extended-char`** objects could signal an error. This bug is now fixed.

11.2.7 Fix for class initargs in delivery

A class now retains its potential initargs during delivery. This fixes a bug that lead to a error of type **`conditions:unknown-keyword-error`** in a CLOS application at delivery level 1.

11.2.8 Fix for Grep Browser

The Grep Browser tool's Process menu would signal an error. This bug is now fixed.

11.2.9 Fixes for Stepper tool

A processor fault in the Stepper tool, seen when continuing after stepping into a function in a different buffer, is now fixed.

An error seen on stepping a **`(lambda ...)`** form (as opposed to **`#'(lambda ...)`**).is now fixed.

11.2.10 Fix for editor paste via menu

The menu command **Paste** has been fixed to delete the selection when used in Macintosh editor / MS Windows emulation.

11.2.10.1 Fix for editor bug with logical pathnames

A bug whereby the editor failed to handle logical pathnames containing hyphens is now fixed.

11.2.11 Fix for bug with small TAGS tables

A bug whereby the editor failed to read a small TAGS file correctly is now fixed.

11.2.12 Fix for editor indentation bug

A bug in the editor's Lisp mode, which caused incorrect indentation in a form where a comment contained an empty string, is now fixed.

11.2.13 Fix for layouts with background color

A 'No color-user found' error seen when closing a CAPI window containing a layout with a background color on Cocoa is now fixed.

11.2.14 Text input choice callbacks

`:callback` was broken for **`capi:text-input-choice`** in LispWorks for Windows 4.4. This bug is now fixed.

11.2.15 Option pane keyboard selection

Selection by keyboard input was broken for **`capi:option-pane`** in LispWorks for Windows 4.4. This bug is now fixed.

11.2.16 Fix for clean-down crash

A bug leading to a crash during **`clean-down`** when large objects exist, often triggered from within the LispWorks IDE on Windows, is now fixed.

11.2.17 Fix for programmatic cursor changes

On Motif, programmatic cursor changes, including those called by some **Preferences...** dialogs in the LispWorks IDE, would cause an error 'Unable to convert :DEFAULT into an XID.'. This bug is now fixed.

11.2.18 Compiler bug with setf names

A bug whereby the compiler failed to recognize **`(setf`** ***symbol*****`)`** as a name is now fixed.

11.2.19 Fix for sqrt and trigonometry functions.

A bug leading to occasional incorrect results returned by **`sqrt`** and trigonometry functions, which could lead to an error in the distributed othello example, is now fixed

11.2.20 Problem inspecting locked hash tables fixed

LispWorks could hang when trying to inspect a hash table during a **`maphash`** operation in a different process. This bug is now fixed.

11.2.21 Safer profiling

Profiling on Windows (using either the REPL or the Profiler tool) has been made safer. Previously, generating a large amount of temporary data during profiling could lead a crash.

11.2.22 Improvement on multi-CPU machines.

LispWorks now gives a better interactive response on multi-CPU Windows machines.

11.2.23 Rich Text on Mac OS X

`capi:rich-text-pane` is implemented on Cocoa but still does not support **`:change-callback`** or printing. This deficiency was omitted from the LispWorks 4.4 documentation.

11.3 User preferences

Despite the change of ownership, LispWorks for Windows 4.4.x still reads and stores user preferences and other registry data in the Windows registry under keys **`HKEY_LOCAL_MACHINE\SOFTWARE\Xanalys\LispWorks`** and **`HKEY_CURRENT_USER\SOFTWARE\Xanalys\LispWorks`**.

For other platforms, user preferences are still read and stored in the directory **`~/.lispworks-config/4.4`**

Therefore on all platforms, if you have previously used Xanalys LispWorks 4.4, your user preferences are retained when using LispWorks 4.4.5.

11.4 New CAPI features

This and the following sections describe the new features and known problems in LispWorks 4.4. They apply to LispWorks 4.4.5 as well.

See the LispWorks *CAPI Reference Manual* for details of all the new CAPI features.

11.4.1 ActiveX controls

The CAPI now supports the embedding of a wide range of third party controls on Windows. For details start with the manual entry for **`capi:ole-control-pane`**.

11.4.2 Cocoa Application Menu support

There is now an interface to the Application menu supporting standard Mac OS X operations in your delivered LispWorks applications.

See the examples in the files:

```
examples/capi/applications/cocoa-application
examples/delivery/macos/simple-application.lisp
examples/delivery/macos/full-application.lisp
```

in the LispWorks for Macintosh library, and the entries in the LispWorks *CAPI Reference Manual*.

11.4.3 Option pane enhancements

`capi:option-pane` now supports separators and individual item enablement.

11.4.4 Toolbar enhancements

`capi:toolbar` now supports buttons with text and no image, and it also now supports enhanced dropdown menus.

11.4.5 Improved focus handling

Callback functions can now be specified to operate on whichever pane within a CAPI interface currently has the input focus. See the description of the initarg **`:callback-type`** in the manual entry for **`capi:callbacks`**.

`capi:text-input-range` and **`capi:option-pane`** now support focus detection by **`capi:pane-has-focus-p`**.

`capi:button` subclasses now respect the **`capi:allows-focus-p`** protocol for controlling tab stops.

11.4.6 Better highlight and cursor behavior in the editor

Highlighting in **`capi:editor-pane`** now extends to the end of each line in the selected region, and not merely to the end of the text on that line as in LispWorks 4.3 and earlier).

Selection now operates when the mouse is dragged outside the window.

The editor cursor is now invisible when the selection will be deleted on input, that is, in MS Windows and Mac OS X editor emulation.

11.4.7 More cursor styles

More system-defined cursor styles are supported for use in **`capi:output-pane`** on Cocoa. See **`capi:simple-pane-cursor`**.

11.4.8 Collection-data callbacks

`capi:collection` subclasses support a new callback type **`:collection-data`**.

11.4.9 Dividers on Mac OS X

Dividers are now implemented on Cocoa. See the manual page for **`capi:row-layout`** or **`capi:column-layout`** for instructions.

Dividers now appear in the Common LispWorks IDE on all platforms.

11.4.10 Rich Text on Mac OS X

`capi:rich-text-pane` is now implemented on Cocoa.

11.4.11 Printing on Mac OS X

Printing for **`capi:output-pane`**, **`capi:editor-pane`**, and in the LispWorks IDE is now implemented on Cocoa, subject to one restriction: **`capi:with-page`** does not work, because Cocoa tries to control page printing.

11.4.12 Mnemonics in grids

capi:grid-layout now supports mnemonics in a title column.

11.4.13 Create callbacks

capi:interface now supports the **:create-callback** initarg, allowing you to run code when the window representation has just been created but before the window actually appears on screen.

11.4.14 Clipboard supports images

The CAPI Clipboard now supports images on Cocoa and Windows. For example:

```
(capi:set-clipboard pane nil nil (list :image image :string "my
image"))
```

where *image* is a **gp:image** object.

11.4.15 Keyboard input specification

Gesture Specs are now documented, and supported in the *input-model* of **capi:output-pane**.

11.4.16 Enter key behavior

On Cocoa and Motif the CAPI now generates **:Kp-Enter** for the **Enter** key (was **:ETX** on Cocoa and a Gesture Spec representing **Return** on Motif). You may need to alter your application's **editor:bind-key** form.

11.4.17 Font control in printing API

The CAPI text printing API (**capi:print-text**, **capi:print-editor-buffer** and **capi:print-file**) now allows the programmer to specify the font.

11.4.18 Improved menu bar behavior with Cocoa dialogs

Menu bar items are now grayed out while a (modal) dialog is on screen. (In LispWorks for Macintosh 4.3, these items incorrectly appeared to be enabled.)

11.5 New graphics ports features

For details see the relevant chapters in the LispWorks *CAPI Reference Manual* and the LispWorks *CAPI User Guide*.

11.5.1 Image access API

New functionality allows you to manipulate images per-pixel.

11.5.2 More image formats supported

`gp:read-external-image` and friends now handle JPEG, PNG, TIFF, GIF and other image types. In previous releases only Bitmaps were supported.

11.6 Objective-C interface on Mac OS X

This new API is documented in the new manual, the LispWorks *Objective-C and Cocoa Interface User Guide and Reference Manual*.

11.6.1 Objective-C functionality

On Mac OS X LispWorks can now:

- Call Objective-C methods
- Implement Objective-C classes and methods
- Be used with a nib file (as created by Apple's Interface Builder).

11.7 SSL interface

`comm:socket-stream` now supports the Secure Sockets Layer via an interface to OpenSSL. See the *LispWorks User Guide* chapter "Socket Stream SSL interface", the relevant entries in the *LispWorks Reference Manual* and also the examples in **`lib/4-4-0-0/examples/ssl/`**.

11.8 IDE improvements

See the *Common LispWorks User Guide* for details of the commands mentioned.

11.8.1 More web browsers supported by Help menu

The LispWorks Help menu can now display the LispWorks HTML documentation using Firefox, Mozilla and Opera browsers as well as Netscape and Internet Explorer.

Note: Reuse of existing browser windows is not supported on all platforms and browsers. In some browsers (Firefox for example) window reuse is an option in the browser.

11.8.2 Improved Profiler tool

The Profiler tool now includes a Call Tree view, in addition to the familiar Cumulative view. The new Call Tree is the default view.

11.8.3 New operations on frames

You can now:

- view the method combination directly from a stack frame by **Debug > Frame > Method Combination**
- enter the Stepper tool directly from a stack frame by **Debug > Frame > Restart Frame Stepping**
- cause a break on return from a stack frame by **Debug > Frame > Break on Return from Frame**

11.8.4 Edit operations in the Find dialog on Cocoa

The Editor tool's **Find...** and **Replace...** dialogs now have an **Edit** menu for clipboard operations on Cocoa. This means that **`Command+C`** copies the selected text, and so on.

11.8.5 Editor cursor improved on Cocoa

The Editor tool now uses an I-beam cursor in Mac OS editor emulation.

11.8.6 Startup problem with international user name

A problem starting the image, often seen by LispWorks for Windows Personal Edition users when there are non-ASCII characters in the user name or machine name, is now fixed.

11.8.7 Setting variable values in the debugger

In the Debugger tool, **Variables > Set...** now evaluates the form entered. **`cl:*`** is bound to the current value of the variable in the frame.

11.8.8 Stepper keys

The following editor commands run the corresponding stepper command in the current Stepper tool. They can be bound to keys using **`editor:bind-key`** to allow the keys to be used when the focus is in the Source or Listener panes of a Stepper tool.

```
Stepper Breakpoint
Stepper Continue
Stepper Macroexpand
Stepper Next
Stepper Restart
Stepper Show Current Source
Stepper Step
Stepper Step Through Call
Stepper Step To Call
Stepper Step To Cursor
Stepper Step To End
Stepper Step To Value
Stepper Undo Macroexpand
```

11.8.9 Breakpoint location

You can locate a breakpoint in your source code via the new **Goto Source** button on the **Edit Breakpoints** dialog.

11.8.10 Improved debugging on Cocoa

Errors in "undebuggable" Cocoa processes can now be investigated through Snapshot Debugging, which makes a copy (the "Snapshot") of the stack and allows use of the Debugger tool, Inspector tool and so on. When this feature is applicable, the **Debug Snapshot** button appears on the error notifier window.

11.9 Other new features

11.9.1 Optimized array access

LispWorks now offers optimal array access for specially constructed Lisp arrays of float or integer types, and for foreign arrays. See the entries for **`system:make-typed-aref-vector`** and **`system:typed-aref`** in the *LispWorks Reference Manual* and **`fli:foreign-typed-aref`** in the LispWorks *Foreign Language Interface User Guide and Reference Manual*.

11.9.2 Fast 32bit integer arithmetic

The new INT32 API allows optimal mod 2^{32} arithmetic in low-safety code. The functionality and performance corresponds to C code using the **`int`** type.

See the Compiler chapter of the *LispWorks User Guide* for details.

11.9.3 Low level stream API

The class **`stream:buffered-stream`** implements the buffer in LispWorks **`file-stream`** and **`comm:socket-stream`**. An API giving programmers access to the stream buffer, already used in CL-HTTP and other user code, is now supported and documented. See the *LispWorks Reference Manual* for details.

11.9.4 Compiler optimizations

The compiler can now produce more efficient floating point code at compiler qualities **`float = 0`** and **`safety = 0`**. Also, floating point multiple values can be optimized (LispWorks 4.3 already optimized floating point arguments).

Code which uses **`multiple-value-bind`** with **`values`** calls is compiled more efficiently now.

11.9.5 Common SQL supports PostgreSQL

Common SQL now includes a native backend for the PostgreSQL database. See the chapter "Common SQL" in the *LispWorks User Guide*.

11.9.6 Regular expressions

You can now do regular expression matching on a string. See the *LispWorks Reference Manual* for documentation on **`lw:find-regexp-in-string`** and **`lw:precompile-regexp`**.

Also, Editor commands **`Find Regexp`** and friends are now documented, in the LispWorks *Editor User Guide*.

11.9.7 MOP changes

`clos:accessor-method-slot-definition` is now AMOP-compliant.

The class **`standard-accessor-method`** now has the AMOP initarg **`:slot-definition`**. The LispWorks 4.3 initarg **`:slot-name`**, and the function **`(setf method-slot-name)`** both no longer exist.

`clos:reader-method-class` and **`clos:write-method-class`** are now called as specified in AMOP.

11.9.8 Win32 console handles international characters

Output to the console on Windows now handles international characters. This fixes breakages when running LispWorks with an international user name or machine name.

Console input now handles international characters, on Windows NT, Windows 2000 and Windows XP only.

11.9.9 User-definable listener commands

You can now define your own commands for use in the listener's top level loop. For details see the manual entry for **`system:define-top-loop-command`** in the *LispWorks Reference Manual.*

11.9.10 Aggregate return values in the FLI

`fli:define-foreign-function` and **`fli:define-foreign-callable`** now support aggregate return values. For details see the description of *result-pointer* in the manual entries in the LispWorks *Foreign Language Interface User Guide and Reference Manual*.

11.9.11 Mousewheel support in LispWorks for Windows

The CAPI and Common LispWorks IDE now respond to Windows **`WM_MOUSEWHEEL`** messages. In previous releases this only worked when third-party software was installed.

11.9.12 Profiler output includes call tree

The macro **`profile`** now outputs a Call Tree, followed by the familiar Cumulative results.

11.9.13 Delivery :keep-clos option

Delivery users specifying the **`:keep-clos`** keyword must review the manual page, as this option now has a new set of better-named values.

11.9.14 top-level-form deprecated

`top-level-form` is deprecated. Please use **`dspec:def`** instead.

11.9.15 Select allows limit

`sql:select` now takes a **`:limit`** argument. See the *LispWorks Reference Manual* for details.

11.9.16 loop module

`loop` is now in a loadable module **`"loop"`** in LispWorks for Windows and LispWorks for Linux. Take care to **`require`** this module at build-time if your application uses evaluated loop forms.

11.9.17 Gesture Spec restriction for cased characters

In order to ensure a consistent representation Gesture Specs no longer support a **`both-case-p`** character combined with the single modifier **`Shift`**. See **`make-gesture-spec`** in the *LispWorks Reference Manual* for details.

11.10 Known Problems

11.10.1 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.

The license serial number and key have to be entered again on the command line if the machine's hostname changes. Some Macintosh machines change hostname when disconnected in the Internet which is the most frequent cause of this problem. See “Failure to start when disconnected from the Internet” on page 83 for a solution.

11.10.2 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 are now the **Get Backtrace** and **Debug Snapshot** buttons so you can obtain a backtrace and debug using a copy of the stack.

The online documentation interface currently starts a new browser window each time for some browsers.

Buttons titled **OK** and **Cancel** (for example **Help > Manuals**) are not descriptive enough.

The Editor's Help interface uses bold to highlight things, but this is not implemented yet and not available in the Monaco font anyway.

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 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. Perhaps LispWorks needs a helper application like the Lisp Monitor in X11/Motif to control this.

The Editor has no keyboard gesture to switch from the Buffers view back to the Text view. Possibly **`Return`** should do this.

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.

The Stepper has no **Edit** menu for pasting text.

There is no support for drag and drop within the tools.

11.10.3 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. Possibly there should also be a way to switch off font smoothing completely.

Toolbars are currently implemented using buttons. This is not necessarily going to be fixed, because CAPI allows toolbars to be placed anywhere on the window whereas Cocoa only allows them at the top.

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 tooltips.

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 now implemented, but note that it works only for the focussed pane.

All dialogs are application modal. This is partly a consequence of having all the GUI running in a single thread.

`capi:docking-layout` doesn't support (un)docking.

There is no meta key in the input-model of **`capi:output-pane`**. Note that, for the editor, 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.

There is no way to make the contents of window flush with the edges and the border around the tools is too large.

Typical Mac (and PC) editors scroll when you drag the mouse above/below the window, but Lispworks only scrolls whilst the mouse is moving.

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.

The editor doesn't allow horizontal scrolling.

In Lisp Mode in the editor, double-clicking on the space immediately after a form selects the following form, even though the green paren matching highlight is indicating the form before the space.

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 **`capi:editor-pane`** and **`capi:tab-layout`** fonts are not good on Motif.

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.

If the default button is placed within a subclass of **`capi:pinboard-layout`**, the background is not drawn when the button pulsates.

The editor displays decomposed characters as separate glyphs.

`capi:multi-column-list-panel` doesn't support sorting by clicking on the column titles. *Update for LispWorks 4.4.5:* this problem is fixed on Mac OS X 10.3, but still does not work on Mac OS X 10.2.

The **`:gap`** option is not supported for the columns of **`capi:multi-column-list-panel`**.

`capi:display-dialog` ignores the specified **`:x`** and **`:y`** coordinates of the dialog (for drop-down sheets the coordinates are not relevant and for dialogs which are separate windows Cocoa forces the window to be in the top-center of the screen).

11.11 Documentation Changes

The Objective-C interface in LispWorks for Macintosh is new and is documented in the LispWorks *Objective-C and Cocoa Interface User Guide and Reference Manual*.

There are now versions of the LispWorks *Editor User Guide*, LispWorks *CAPI User Guide* and LispWorks *KnowledgeWorks and Prolog User Guide* specifically for the Cocoa GUI. Users of the X11 / Motif GUI may prefer to obtain alternate versions of these manuals from www.lispworks.com.

11.12 Binary Incompatibilty

If you have binaries (fasl files) which were compiled using LispWorks 4.3 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 4.4.5.

However, code already compiled with LispWorks 4.4.0 is compatible with this patched release LispWorks 4.4.5.