LispWorks®

Release Notes and Installation Guide

Version 6.0

Copyright and Trademarks

LispWorks Release Notes and Installation Guide

Version 6.0

December 2009

Address

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

Telephone

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

From elsewhere: +44 1223 421860

Fax

From North America: 617 812 8283

From elsewhere: +44 870 2206189

www.lispworks.com

Contents

1

Introduction

1.1 LispWorks Editions

LispWorks is available in three product editions on the Mac OS X, Windows, Linux, x86/x64 Solaris and FreeBSD platforms.

The main differences between the editions are outlined below. Further information about the LispWorks Editions can be found at `www.lispworks.com/products`

Note: on SPARC Solaris and HP-UX LispWorks is licensed differently to other platforms, as detailed in “LispWorks for UNIX” on page 2.

1.1.1 Personal Edition

LispWorks Personal Edition allows you to explore a fully enabled Common Lisp programming environment and to develop small- to medium-scale programs for personal and academic use. It includes:

- Native graphical IDE
- Full Common Lisp compiler
- COM/Automation API on Microsoft Windows

LispWorks Personal Edition has several limitations designed to prevent commercial exploitation of this free product. These are:

- A heap size limit
- A time limit of 5 hours for each session.
- 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.

LispWorks Personal Edition has no license fee. Download it from

www.lispworks.com/downloads.

1.1.2 Professional Edition

LispWorks 6.0 Professional Edition includes:

- Fully supported commercial product
- Delivery of commercial end-user applications and libraries
- CLIM 2.0 on X11 / Motif and Windows
- 30-day free “Getting Started” technical support

1.1.3 Enterprise Edition

LispWorks 6.0 Enterprise Edition provides further support for the software needs of the modern enterprise, including:

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

1.2 LispWorks for UNIX

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

LispWorks 6.0 for UNIX is available with a basic developer license, and the add-on products CLIM, KnowledgeWorks, LispWorks ORB and Application Delivery are each separately available.

1.3 Further details

For further information about LispWorks products visit

www.lispworks.com

To purchase LispWorks please follow the instructions at:

www.lispworks.com/buy

1.4 About this Guide

This document is an installation guide and release notes for LispWorks 6.0 on Mac OS X, Windows, Linux, x86/x64 Solaris, FreeBSD, SPARC Solaris and HP-UX 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

Chapters 2-7 explain in brief and sufficient terms how to complete a LispWorks installation on Mac OS X, Windows, Linux, x86/x64 Solaris, FreeBSD, or UNIX (meaning HP-UX or SPARC/Solaris). Choose the chapter for your platform: Chapter 2, “Installation on Mac OS X”, Chapter 3, “Installation on Windows”, Chapter 4, “Installation on Linux”, Chapter 5, “Installation on x86/x64 Solaris”, Chapter 6, “Installation on FreeBSD” or Chapter 7, “Installation on UNIX”.

Chapters 8-11 explain in detail everything necessary to configure, run, and test LispWorks 6.0. Choose the chapter for your platform: Chapter 8, “Configuration on Mac OS X”, Chapter 9, “Configuration on Windows”, Chapter 10, “Configuration on Linux, x86/x64 Solaris, and FreeBSD” or Chapter 11, “Configuration 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 12, “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 13, “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 6.0 for Macintosh. 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.

2.1 Choosing the Graphical User Interface

LispWorks for Macintosh supports three different graphical interfaces. Most users choose the native Mac OS X GUI, but you can use X11 GUI option instead, which supports both GTK+ and Motif. (Motif is deprecated, though.)

Different executables and supporting files are supplied for the two GUI options. You need to decide at installation time which of these you will use, or you can install support for both. If you install just one GUI option and later decide to install the other, 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 formats: HTML and PDF. You can chose whether to install it as described in Section 2.4, “Installing LispWorks for Macintosh”.

The HTML format can be used from within the LispWorks IDE via the **Help** menu. You will need to have a suitable web browser installed. You can also reach the HTML documentation via the alias **`LispWorks 6.0/HTML Documentation.htm`**. If you choose not to install the documentation, you will not be able to access the HTML Documentation from the LispWorks **Help** menu.

The PDF format 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`**. The simplest way to locate these PDF files is the alias **`LispWorks 6.0/PDF Documentation`**. To view and print these files, you will need a PDF viewer such as Preview (standard on Mac OS X) or Adobe® Reader® (which can be downloaded from the Adobe website at **`www.adobe.com`**).

2.3 Software and hardware requirements

LispWorks 6.0 is a universal binary, which supports Macintosh computers containing either PowerPC or Intel CPUs.

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

Table 2.1 System requirements on Mac OS X

Product	Hardware Requirements	Software Requirements
LispWorks (32-bit) for Macintosh	Intel or G3/G4/G5 processor. 240MB of disk space including documentation.	Mac OS X version 10.3.x, 10.4.x, 10.5.x or 10.6.x GTK+ 2.4 or higher if you want to run the GTK+ GUI. Open Motif 2.3 and Imlib if you want to run the deprecated Motif GUI.
LispWorks (64-bit) for Macintosh	Intel or G5 processor. 285MB of disk space including documentation	Mac OS X version 10.5.x or 10.6.x. GTK+ 2.4 or higher if you want to run the GTK+ GUI. Open Motif 2.3 and Imlib if you want to run the deprecated Motif GUI.

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 6.0. There may be a downloadable patch bundle which upgrades LispWorks to version 6.0.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 6.0.

2.4.2 Information for Beta testers

Users of LispWorks 6.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.0. You can run the Beta installer and select the **Uninstall** option (and then remove any patches) or simply drag the `LispWorks 6.0` folder to the trash.

2.4.3 Information for users of previous versions

You can install LispWorks 6.0 in the same location as LispWorks 5.1 or previous versions. If you always choose the default install location, a new `LispWorks 6.0` folder will be created alongside the other versions.

Similarly LispWorks Personal Edition 6.0 can be installed in the same location as previous versions.

2.4.4 Use an adminstrator account

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

However, a non-administrator may install LispWorks elsewhere.

2.4.5 Launch the LispWorks installer

If you have downloaded LispWorks, you may need to mount the disk image containing the installer. This is called `LispWorks-6.0.dmg` or `LispWorks64bit-6.0.dmg` — simply double-click on the `.dmg` file to mount it.

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

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

To install LispWorks (64-bit) for Macintosh, open the `macos64` folder and double-click on the `LispWorks64bit_Installer` application to launch it.

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

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

2.4.8 Select Destination

All the files installed with LispWorks are placed in the LispWorks folder, which is named **`LispWorks 6.0`**, **`LispWorks 6.0 (64-bit)`** or **`LispWorks Personal 6.0`** depending on which edition you are installing. By default, the LispWorks folder is placed in the main **`Applications`** folder but you can choose an alternative location during installation by clicking the **Select Folder...** button.

Click **Continue** after selecting a folder.

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

2.4.9 Choose your installation type

Choose the native Mac OS X GUI and/or the X11 GUI option.

Different executables and supporting files are supplied for the two GUI options. If you install just one of these and later decide to install the other, you can simply run the installer again.

2.4.9.1 The native Mac OS X GUI

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

2.4.9.2 The X11 GTK+ and Motif GUIs

If you want to use LispWorks with either of the alternative X11 GUIs, choose **Custom Install** and select the option "LispWorks with X11 IDE".

The default X11 GUI is GTK+. Motif is also available, but is deprecated. You can select Motif at runtime.

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

- An X server such as Apple's X11.app, available at **`www.apple.com`**, and
- one of GTK+ 2.4 (or higher) or Open Motif 2.3.

If you use Open Motif, you will also need Imlib (but not Imlib2). Imlib version 1.9.13 or later is recommended.

None of these are required at the time you install LispWorks, however.

The X11 GUIs are 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.

2.4.10 Installing and entering license data

Now click **Install**.

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

Your license key will be supplied to you in email from Lisp Support.

If you have problems with your LispWorks license key, send it to **`lisp-keys@lispworks.com`**, describing what happens after you enter it, preferably with a screenshot.

Note: the LispWorks Personal Edition installer does not ask you to enter license data.

2.4.11 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**.

Note: LispWorks may not be visible in the Dock until you restart the computer or log out and then log back in.

2.4.12 Finishing up

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

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

2.4.13 Installing Patches

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

2.4.14 Obtaining X11 GTK+

LispWorks does not provide GTK+ libraries, so you need to install third-party libraries, such as

- the gtk+2 package from the Fink Project at `www.finkproject.org`, or
- the gtk2 package from MacPorts at `www.macports.org`

Note: you need the x11 gtk2 libraries, not GTK-OSX (Quartz).

2.4.15 Obtaining Open Motif and Imlib

LispWorks 6.0 for Macintosh on X11/Motif requires Open Motif 2.3 and Imlib.

The Open Motif library for 32-bit LispWorks is `/usr/local/lib/libXm.4.dylib`.

Lisp Support can supply suitable Motif and Imlib libraries if you need them.

Note: The Motif GUI is deprecated. A GTK+ GUI is available.

2.5 Starting LispWorks for Macintosh

2.5.1 Start the native Mac OS X LispWorks GUI

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

Note: The LispWorks folder is described in "Select Destination" on page 9.

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

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

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

and pass it to the supplied `lispworks-6-0-0-macos-universal` image.

See Section 8.3, "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-6-0-0-macos-universal`, and `save-image` is not available.

2.5.2 Start the GTK+ LispWorks GUI

Assuming you have installed the "LispWorks with X11 IDE" option, and that you have X11 running and GTK+ installed, you can now start LispWorks with the GTK+ GUI.

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

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

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

```
xterm% cd "/Applications/LispWorks 6.0"
xterm% ./lispworks-6-0-0-macos-universal-gtk
LispWorks(R): The Common Lisp Programming Environment
Copyright (C) 1987-2009 LispWorks Ltd.  All rights reserved.
Version 6.0.0
Saved by LispWorks as lispworks-6-0-0-x86-darwin-gtk, at 04 Sep
2009 22:09
User dubya on octane
; Loading text file /Applications/LispWorks 6.0/Library/lib/6-0-
0-0/config/siteinit.lisp
; Loading text file /Applications/LispWorks 6.0/Library/lib/6-0-
0-0/private-patches/load.lisp
; Loading text file /u/ldisk/dubya/.lispworks

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

  The LispWorks GTK+ IDE should appear.

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

- 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 6.0"
xterm% ./lispworks-6-0-0-macos-universal-gtk -env
```

  The LispWorks GTK+ IDE should appear.

- 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 save-image script that calls

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

and run the supplied **lispworks-6-0-0-macos-universal-gtk** image with **-build** followed by the script filename on its command line.

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

See Section 8.3, “Configuring your LispWorks installation” for more information about configuring your LispWorks image for your own needs.

2.5.3 Start the Motif LispWorks GUI

Assuming you have installed the "LispWorks with X11 IDE" option, and that you have X11 running and Motif installed, you can use LispWorks with the Motif GUI.

You first must load the Motif GUI into the supplied **lispworks-6-0-0-macos-universal-gtk** image, by

```
(require "capi-motif")
```

This loads the necessary module and makes Motif the default library for CAPI.

Then you can start the LispWorks IDE by calling **env:start-environment** as shown in “Start the GTK+ LispWorks GUI” on page 13. You might want to save an image with the **"capi-motif"** module pre-loaded: do this with a save-image script containing

```
(require "capi-motif")
```

2.6 Upgrading to LispWorks Enterprise Edition

You can upgrade from Professional Edition to LispWorks (32-bit) Enterprise by doing **Help > Register...** and entering your Enterprise license key.

To upgrade to 64-bit LispWorks, contact

lisp-sales@lispworks.com

3

Installation on Windows

This chapter is an installation guide for LispWorks 6.0 for Windows and LispWorks 6.0 (64-bit) for Windows. Chapter 9 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

3.1 Documentation

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

If you install the HTML documentation, then it can be used from within the the LispWorks IDE via the **Help** menu. It is also available from the Windows **Start** menu under **Start > All Programs > LispWorks 6.0 > HTML Documentation**.

The PDF format is suitable for printing. Each manual in the documentation set is presented in a separate PDF file, available from the **Start** menu under **Start > All Programs > LispWorks 6.0 > PDF Documentation**. To view and print these files, you will need a PDF viewer such as Adobe® Reader®. If you do not already have this, it can be downloaded from the Adobe website.

3.2 Installing LispWorks for Windows

3.2.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as an installer containing version 6.0. There may be a downloadable patch bundle which upgrades LispWorks to version 6.0.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 6.0.

3.2.2 Visual Studio runtime components and Windows Installer

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

It will also automatically install Windows Installer 3.1 when needed.

3.2.3 Installing over previous versions

You can install LispWorks 6.0 in the same location as LispWorks 5.1, LispWorks 5.0 or LispWorks 4.4.5. This is the default installation location.

You can also install LispWorks 6.0 without uninstalling older versions such as Xanalys LispWorks 4.4 or Xanalys LispWorks 4.3 provided that the chosen installation directory is different.

The LispWorks Personal Edition installation behaves in the same way.

3.2.4 Information for Beta testers

Users of LispWorks 6.0 Beta should completely uninstall it before installing LispWorks 6.0. Remember to remove any patches added since the Beta release.

3.2.5 To install LispWorks

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

To install LispWorks (64-bit) for Windows run
x64-windows\LispWorks64bit_Installer.exe.

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

3.2.5.1 Entering the License Data

Enter your serial number and license key when the installer asks for these details in the **Customer Information** screen.

Your license key will be supplied to you in email from Lisp Support.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, describing what happens after you enter it.

3.2.5.2 Installation location

By default LispWorks installs in All Users space in
C:\Program Files\LispWorks

To install LispWorks in a non-default location (for example, to ensure it is accesible only by the licensed user on a multi-user system such as a login server (remote desktop)), select **Custom** setup in the **Setup Type** screen. Then click **Change...** in the **Custom Setup** screen and choose the desired location in the **Change Current Destination Folder** dialog. Do not simply move the LispWorks folder later, as this will break the installation.

3.2.5.3 Installing the Documentation

By default all the documentation is installed.

If you do not want to install the HTML Documentation, select **Custom** setup in the **Setup Type** screen and select **This feature will not be available** in the HTML Documentation feature in the **Custom Setup** screen.

You can also choose not to install the PDF Documentation, in a similar way.

You can add the HTML Documentation and the PDF Documentation later, by re-running the installer. The documentation is also available at **www.lisp-works.com/documentation**.

3.2.5.4 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.

3.2.5.5 Starting LispWorks

When the installation is complete, you can start LispWorks by choosing **Start > All Programs > LispWorks 6.0 > LispWorks**.

Note: After installation you must not move or copy the LispWorks folder, since the system records the installation location. Moreover 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.

3.3 Upgrading to LispWorks Enterprise Edition

You can upgrade from Professional Edition to LispWorks (32-bit) Enterprise by doing **Help > Register...** and entering your Enterprise license key.

To upgrade to 64-bit LispWorks, contact

lisp-sales@lispworks.com

4

Installation on Linux

This chapter is an installation guide for LispWorks 6.0 for Linux and LispWorks 6.0 (64-bit) for Linux. Chapter 10, 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
155MB of disk space for Enterprise Edition (32-bit) plus documentation	Any distribution with glibc 2.6.9 or later.

Table 4.1 System requirements on Linux

Hardware Requirements	Software Requirements
175MB of disk space for Enterprise Edition (64-bit) plus documentation	GTK+ 2.4 or higher to run the GTK+ GUI. Open Motif 2.2.x and Imlib to run the deprecated Motif GUI
Any modern machine is likely to have sufficient RAM to run LispWorks as distributed.	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 4.1 System requirements on Linux

4.1.1 GUI libraries

LispWorks 6.0 for Linux requires that the X11 release 6 (or higher) is installed. It also requires that either GTK+ or Open Motif with Imlib are installed.

The remainder of this section contains the details for each of these distinct GUI options.

4.1.1.1 GTK+

In order for the LispWorks IDE to run “out of the box”, GTK+ must be installed on the target machine.

GTK+ version 2.4 or higher is required.

4.1.1.2 Motif

Open Motif version 2.2 or higher is required to run LispWorks with the Motif GUI.

Download and install Open Motif 2.2.x from your Linux distribution or from **`www.motifzone.net`**. Your systems administrator may be able to help if you do not know how to do this.

You will also need Imlib (not Imlib2). Install this from your Linux distribution. Imlib version 1.9.13 or later is recommended.

Note: You should be able to run the LispWorks 6.0 Motif GUI and LispWorks 5.x simultaneously with Open Motif installed.

4.1.2 Disk requirements

To install without documentation and optional modules, 32-bit LispWorks requires about 45MB and 64-bit LispWorks requires about 60MB. Installing the documentation adds about 110MB and the optional modules about 15MB. A full installation of the 64-bit Enterprise Edition with all documentation and optional modules requires about 185MB.

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

4.2 License agreement

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

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

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

Now run the one of following scripts.

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

- For the Professional and Enterprise Editions, run

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

- For the Personal Edition, run:

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

Enter “yes” if you agree to the license terms.

4.3 Software on the CD-ROM

LispWorks 6.0 for Linux is supplied as a download or on a CD-ROM. There are two different formats: RedHat Package Management (RPM) files and **tar** files. RPM is a utility like **tar**, except it can actually install products after unpacking them. See Section 4.4.3 for more information. Both formats are in the **x86-linux** and **amd64-linux** directories on our ftp server or on your CD-ROM.

4.3.1 Professional and Enterprise Edition distributions

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

The RPM package name for the Professional/Enterprise Edition is **lispworks**.

For the Professional Edition the separately installable packages are:

```
lispworks-clim
```

and for the Enterprise Edition the separately installable packages are:

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

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

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 6.0. There may be a downloadable patch bundle which upgrades LispWorks to version 6.0.x. You need to complete the main installa-

tion before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 6.0.

4.4.2 Information for Beta testers

Users of LispWorks 6.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.0.

See “Uninstalling LispWorks for Linux” on page 32 for instructions.

4.4.3 Installation from the binary RPM file

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

If you already have LispWorks 6.0 Beta installed, please uninstall it before installing this product. See Section 4.9, “Uninstalling LispWorks for Linux”.

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

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

try upgrading to RPM 3.0.2 or greater.

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

4.4.3.1 Installation directories

By default 32-bit LispWorks is installed in **`/usr/lib/LispWorks`** and a symbolic link to the executable is placed in **`/usr/bin/lispworks-6-0-0-x86-linux`**. Similarly, 64-bit LispWorks is installed in **`/usr/lib64/LispWorks`** and a symbolic link to the executable is placed in **`/usr/bin/lispworks-6-0-0-amd64-linux`**. However, the RPM is relocatable, and the **`--prefix`** option can be used to allow the installation of LispWorks in a non-default directory. The default prefix is **`/usr`**.

Note: RPM version 4.2 has a bug which can hinder secondary installations (CLIM, Common SQL, LispWorks ORB or KnowledgeWorks) in a user-specified directory. See "RPM_INSTALL_PREFIX not set" on page 100 for a workaround.

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

4.4.3.2 Selecting the correct RPM files

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

`lispworks-6.0-`***n*****`.`*****arch*****`.rpm`**

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

Note: For the Personal Edition, use `lispworks-personal-6.0-*.i386.rpm` wherever `lispworks-6.0-*.i386.rpm` is mentioned in this document. See Section 1.1.1, "Personal Edition" for more information specific to the Personal Edition.

4.4.3.3 Installing or upgrading LispWorks for Linux

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

1. Locate the RPM installation file `lispworks-6.0-`***n*****`.i386.rpm`**.
2. Install or upgrade LispWorks in the standard RPM way, for example:

   `rpm --install lispworks-6.0-`***n*****`.i386.rpm`**

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

   `rpm --install --prefix <`***directory*****`> lispworks-6.0-`*****n*****`.i386.rpm`**

   installs LispWorks in <*directory*>.

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

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

See Section 4.6 for instructions on entering your license details.

4.4.3.4 Installing CLIM 2.0

The following module is packaged as a separate RPM file for installation after the main `lispworks` package. It is available in all LispWorks Editions except the Personal Edition.

File Distribution	Layered Product
`lispworks-clim-6.0-`***n***`.i386.rpm`	CLIM 2.0

Table 4.2 File distributions for layered products in all Editions other than Personal

Install this module if required by substituting the above filename into the same commands you used to install the LispWorks package (Section 4.4.3.3).

If you used a `--prefix` argument when installing LispWorks, then use the same prefix for this module.

4.4.3.5 Installing loadable Enterprise Edition modules

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

File Distribution	Layered Product
`lispworks-clim-6.0-`***n***`.i386.rpm`	CLIM 2.0

Table 4.3 File distributions for layered products in the Enterprise Edition

File Distribution	Layered Product
lispworks-kw-6.0-*n*.i386.rpm	KnowledgeWorks
lispworks-corba-6.0-*n*.i386.rpm	LispWorks ORB
lispworks-sql-6.0-*n*.i386.rpm	Common SQL

Table 4.3 File distributions for layered products in the Enterprise Edition

Install these modules as described in Section 4.4.3.4.

4.4.3.6 Documentation and saving space

Documentation in HTML and PDF format is provided with all editions. PostScript format is available to download. To obtain copies of the printable manuals, see Section 4.8, “Printable LispWorks documentation”.

Documentation is installed by default in the **lib/6-0-0-0/manual** sub-directory of the LispWorks installation directory.

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

```
rpm --install --excludedocs --prefix <directory>
lispworks-6.0-n.i386.rpm
```

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

```
rpm --install --prefix <directory> --replacepkgs
lispworks-6.0-n.i386.rpm
```

4.4.3.7 Installing Patches

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

4.4.4 Installation from the tar files

The LispWorks distribution is also provided as `tar` files compressed using `gzip` for use if you do not have an appropriate version of RPM to unpack the RPM binary file. The gzipped files for 32-bit LispWorks are as follows:

Table 4.4 Files for 32-bit Professional and Enterprise Editions

`lw60-x86-linux.tar.gz`	32-bit LispWorks image, modules and examples
`lwdoc60-x86-linux.tar.gz`	Documentation in HTML and PDF formats

Note: The gzipped files for LispWorks Personal Edition and LispWorks (64-bit) Enterprise Edition have similar names.

To install from these files:

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

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

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

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

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

See Section 4.6 for how to enter your license details.

4.4.4.1 Installing Patches

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

4.5 LispWorks looks for a license key

If you installed the Professional or Enterprise 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 10.2, “License keys”.

4.6 Running LispWorks

The LispWorks executable is located in **/usr/lib/LispWorks** or **/usr/lib64/LispWorks** directory of the installation (assuming the default prefix of **/usr**) and should not be moved without being resaved because it needs to be able to locate the corresponding library directory on startup. There is also a symbolic link from the **/usr/bin** directory.

The LispWorks executable is named as shown here:.

lispworks-personal-6-0-0-x86-linux	Personal Edition
lispworks-6-0-0-x86-linux	32-bit Professional or Enterprise Edition
lispworks-6-0-0-amd64-linux	64-bit Enterprise Edition

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 97 for details if this does not happen.

4.6.1 Entering the license data

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

lispworks-6-0-0-x86-linux --lwlicenseserial *SERIALNUMBER* --lwlicensekey *LICENSEKEY*

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

```
LispWorks license installed successfully.
```

should be printed and thereafter you can run LispWorks without those command line arguments.

Your license key will be supplied to you in email from Lisp Support.

If you have problems with your LispWorks license key, send it to **lispkeys@lispworks.com**, describing what happens after you enter it.

4.7 Configuring the image

If you installed the Professional or Enterprise 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 10, “Configuration on Linux, x86/x64 Solaris, and FreeBSD”.

4.8 Printable LispWorks documentation

In a default Professional/Enterprise installation, the **lib/6-0-0-0/manual/offline** directory contains PDF format versions of the manuals.

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

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

4.9 Uninstalling LispWorks for Linux

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

```
rpm --erase lispworks-6.0
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named `lispworks-patches6.0`. The same applies to additional RPM packages such as `lispworks-corba`.

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

If you installed LispWorks from the tar archives, simply do

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

4.10 Upgrading to LispWorks Enterprise Edition

You can upgrade from Professional Edition to LispWorks (32-bit) Enterprise by doing **Help > Register...** and entering your Enterprise license key.

To upgrade to 64-bit LispWorks, contact

```
lisp-sales@lispworks.com
```

5

Installation on x86/x64 Solaris

This chapter is an installation guide for LispWorks 6.0 for x86/x64 Solaris. Chapter 10, discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

5.1 Software and hardware requirements

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

Hardware Requirements	Software Requirements
For 32-bit LispWorks, 130MB of disk space	Solaris 10 (release 5/08 or later) or OpenSolaris (release 2009.06 or later)

Table 5.1 System requirements on x86/x64 Solaris

Hardware Requirements	Software Requirements
For 64-bit LispWorks, 140MB of disk space	GTK+ 2.4 or higher to run the GTK+ GUI. Motif 2.1 and Imlib to run the deprecated Motif GUI
Any modern machine is likely to have sufficient RAM to run LispWorks as distributed.	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 5.1 System requirements on x86/x64 Solaris

5.1.1 GUI libraries

LispWorks 6.0 for x86/x64 Solaris requires that the X11 release 6 (or higher) is installed. It also requires that either GTK+ or Motif with Imlib are installed.

The remainder of this section contains the details for each of these distinct GUI options.

5.1.1.1 GTK+

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

GTK+ version 2.4 or higher is required.

5.1.1.2 Motif

Motif 2.1 or higher is required to run LispWorks with the Motif GUI.

The Motif libraries are installed as part of the SUNWmfrun package. It is usually preinstalled on Solaris 10 and is available for download from Sun for OpenSolaris.

You will also need Imlib (not Imlib2). Imlib version 1.9.13 or later is recommended. Contact Lisp Support if you need this.

5.1.2 Disk requirements

32-bit LispWorks requires about 130MB to install.

64-bit LispWorks requires about 140MB to install.

The installation includes about 70MB of documentation.

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

5.2 Software on the CD-ROM

LispWorks 6.0 for x86/x64 Solaris is supplied as a standard package file to download or on your CD-ROM. There are two variants, so be sure to choose the one for which you have purchased a license:

32-bit LispWorks is in the **x86-solaris** directory.

64-bit LispWorks is in the **amd64-solaris** directory.

5.2.1 Professional and Enterprise Edition distributions

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

5.2.2 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 for the Personal Edition is **LispWorksPersonal60-32bit**.

5.3 Installing LispWorks for x86/x64 Solaris

5.3.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as a standard software package file containing version 6.0. There may be a downloadable

patch bundle which upgrades LispWorks to version 6.0.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 6.0.

5.3.2 Information for Beta testers

Users of LispWorks 6.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.0.

See "Uninstalling LispWorks for x86/x64 Solaris" on page 39 for instructions.

5.3.3 Installation directories

32-bit LispWorks is installed by default in **`/opt/LispWorks/lib/LispWorks`** and a symbolic link to the executable is placed in **`/opt/LispWorks/bin/lispworks-6-0-0-x86-solaris`**.

64-bit LispWorks is installed by default in **`/opt/LispWorks/lib/amd64/LispWorks`** and a symbolic link to the executable is placed in **`/opt/LispWorks/bin/lispworks-6-0-0-amd64-solaris`**.

LispWorks Personal Edition is installed by default in **`/opt/LispWorks/lib/LispWorksPersonal`** and a symbolic link to the executable is placed in **`/opt/LispWorks/bin/lispworks-personal-6-0-0-x86-solaris`**.

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.

5.3.4 Selecting the correct software package file

The LispWorks (32-bit) Professional/Enterprise software package file is called **`LispWorks60-32bit`** and can be found in the **`x86-solaris`** directory of the LispWorks 6.0 CD-ROM. The package name is **`LispWorks60-32bit`**.

The LispWorks (64-bit) Enterprise software package file is called **`LispWorks60-64bit`** and can be found in the **`amd64-solaris`** directory of the LispWorks 6.0 CD-ROM. The package name is **`LispWorks60-64bit`**.

The Personal Edition software package file is called **LispWorksPersonal60-32bit**. The package name is **LispWorksPersonal60-32bit**.

Note: the software may be supplied in a compressed format with a **.gz** extension. Uncompress it using **gunzip**.

5.3.5 Installing the package file

To install LispWorks, perform the following steps as root:

1. Locate the software package file.
2. Install or upgrade LispWorks in the standard way, for example:

```
pkgadd -d LispWorks60-32bit all
```

for 32-bit LispWorks, or

```
pkgadd -d LispWorks60-64bit all
```

for 64-bit LispWorks.

3. The license terms are presented. Enter "yes" if you agree to them.

See Section 5.5 for instructions on entering your license serial number and key.

5.3.6 Installing Patches

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

5.4 LispWorks looks for a license key

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

For instructions on entering your license key, see Section 5.5.1, "Entering the license data" below.

For more information about license keys, see Section 10.2, "License keys".

5.5 Running LispWorks

The LispWorks executable is located as shown here:

/opt/LispWorks/lib/LispWorksPersonal	Personal Edition
/opt/LispWorks/lib/LispWorks	Professional or Enterprise Edition
/opt/LispWorks/lib/amd64/LispWorks	64-bit Enterprise Edition

This executable should not be moved without being resaved because it needs to be able to locate the corresponding library directory on startup.

The LispWorks executable is named as shown here:.

lispworks-personal-6-0-0-x86-solaris	Personal Edition
lispworks-6-0-0-x86-solaris	Professional or Enterprise Edition
lispworks-6-0-0-amd64-solaris	64-bit Enterprise Edition

There is also a symbolic link from the **/opt/LispWorks/bin** directory.

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 97 for details if this does not happen.

5.5.1 Entering the license data

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

lispworks-6-0-0-x86-solaris --lwlicenseserial *SERIALNUMBER* --lwlicensekey *LICENSEKEY*

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

```
LispWorks license installed successfully.
```

should be printed and thereafter you can run LispWorks without those command line arguments.

Your license key will be supplied to you in email from Lisp Support.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, describing what happens after you enter it.

5.6 Configuring the image

If you installed the Professional or Enterprise 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 10, "Configuration on Linux, x86/x64 Solaris, and FreeBSD".

5.7 Printable LispWorks documentation

In a default Professional/Enterprise installation, the **lib/6-0-0-0/manual/offline** directory contains PDF format versions of the manuals.

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

5.8 Uninstalling LispWorks for x86/x64 Solaris

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

```
pkgrm -n LispWorks60-32bit
```

or

```
pkgrm -n LispWorks60-64bit
```

5.9 Upgrading to LispWorks Enterprise Edition

You can upgrade from Professional Edition to LispWorks (32-bit) Enterprise by doing **Help > Register...** and entering your Enterprise license key.

To upgrade to 64-bit LispWorks, contact

```
lisp-sales@lispworks.com
```

6

Installation on FreeBSD

This chapter is an installation guide for LispWorks 6.0 for FreeBSD. Chapter 10, discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

6.1 Software and hardware requirements

An overview of system requirements is provided in Table 6.1. The sections that follow discuss any relevant details.

Hardware Requirements	Software Requirements
160MB of disk space for Enterprise Edition plus documentation	FreeBSD 7.x, or later with compat7x (if you want to run on FreeBSD 6.x, then please contact Lisp Support)

Table 6.1 System requirements on FreeBSD

Hardware Requirements	Software Requirements
Any modern machine is likely to have sufficient RAM to run LispWorks as distributed.	GTK+ 2.4 or higher to run the GTK+ GUI. Open Motif 2.2.x and Imlib to run the deprecated Motif GUI
	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 6.1 System requirements on FreeBSD

6.1.1 GUI libraries

LispWorks 6.0 for FreeBSD requires that the X11 release 6 (or higher) is installed.

LispWorks 6.0 also requires that either GTK+ or Open Motif with Imlib are installed.

The remainder of this section contains the details for each of these distinct GUI options.

6.1.1.1 GTK+

In order for the LispWorks IDE to run “out of the box”, GTK+ must be installed on the target machine.

GTK+ version 2.4 or higher is required.

6.1.1.2 Motif

Open Motif version 2.2 or higher is required to run LispWorks with the Motif GUI.

Install Open Motif 2.2.x from the FreeBSD distribution or ports tree. Your systems administrator may be able to help if you do not know how to do this.

You will also need Imlib (not Imlib2). Install this from the FreeBSD distribution or ports tree. Imlib version 1.9.13 or later is recommended.

6.1.2 Disk requirements

LispWorks requires about 160MB to install, which includes 110MB of documentation.

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

6.2 License agreement

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

You must run this script as the same user that later performs the installation.

- For the Professional and Enterprise Editions, run

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

- For the Personal Edition, run:

  ```
  sh lwfper-license.sh
  ```

Enter “yes” if you agree to the license terms.

6.3 Software on the CD-ROM

LispWorks 6.0 for FreeBSD is supplied as a standard package file in the **`x86-freebsd`** directory on your CD-ROM.

6.3.1 Professional and Enterprise Edition distributions

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

6.4 Installing LispWorks for FreeBSD

6.4.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as a standard software package file containing version 6.0. There may be a downloadable patch bundle which upgrades LispWorks to version 6.0.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

6.4.2 Information for Beta testers

Users of LispWorks 6.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.0.

See "Uninstalling LispWorks for FreeBSD" on page 47 for instructions.

6.4.3 Installation directories

By default LispWorks is installed in **`/usr/local/lib/LispWorks`** and a symbolic link to the executable is placed in **`/usr/local/bin/lispworks-6-0-0-x86-freebsd`**. However, the software package is relocatable, and the **`-p`** option can be used to allow the installation of LispWorks in a user-specified directory. The default prefix is **`/usr/local`**.

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

6.4.4 Selecting the correct software package file

The LispWorks Professional/Enterprise software package file is called

```
lispworks-6.0.tgz
```

and can be found in the **`x86-freebsd`** directory of the LispWorks 5.1 CD-ROM.

The Personal Edition software package file is called

```
lispworks-personal-6.0.tgz
```

6.4.5 Installing LispWorks for FreeBSD

To install LispWorks, perform the following steps as root:

1. Locate the software package file.
2. Install or upgrade LispWorks in the standard way, for example:

   ```
   pkg_add lispworks-6.0.tgz
   ```

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

   ```
   pkg_add -p <directory> lispworks-6.0.tgz
   ```

   installs LispWorks in <*directory*>.

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

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

See Section 6.6 for instructions on entering your license details.

6.4.6 Installation by non-root users

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

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

```
pkg_add -p installation-directory -R lispworks-6.0.tgz
```

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

6.4.7 Installing Patches

After completing the main installation of the Professional or Enterprise Edition, ensure you install the latest patches from the package file available for

download at **www.lispworks.com/downloads/patch-selection.html**. Patch installation instructions are in the README file accompanying the patch download.

6.5 LispWorks looks for a license key

If you installed the Professional or Enterprise 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 6.6.1, "Entering the license data" below.

For more information about license keys, see Section 10.2, "License keys".

6.6 Running LispWorks

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

The LispWorks executable is named as shown here:.

lispworks-personal-6-0-0-x86-freebsd	Personal Edition
lispworks-6-0-0-x86-freebsd	Professional or Enterprise Edition

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See "Troubleshooting" on page 97 for details if this does not happen.

6.6.1 Entering the license data

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

```
lispworks-6-0-0-x86-freebsd --lwlicenseserial SERIALNUMBER
--lwlicensekey LICENSEKEY
```

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

```
LispWorks license installed successfully.
```

should be printed and thereafter you can run LispWorks without those command line arguments.

Your license key will be supplied to you in email from Lisp Support.

If you have problems with your LispWorks license key, send it to `lisp-keys@lispworks.com`, describing what happens after you enter it.

6.7 Configuring the image

If you installed the Professional or Enterprise 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 10, "Configuration on Linux, x86/x64 Solaris, and FreeBSD".

6.8 Printable LispWorks documentation

In a default Professional/Enterprise installation, the `lib/6-0-0-0/manual/offline` directory contains PDF format versions of the manuals.

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

6.9 Uninstalling LispWorks for FreeBSD

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

```
pkg_delete lispworks-6.0
```

If patches have been installed, then you will first need to uninstall that package, which will be named `lispworks-patches6.0`.

6.10 Upgrading to LispWorks Enterprise Edition

You can upgrade from Professional Edition to LispWorks (32-bit) Enterprise by doing **Help > Register...** and entering your Enterprise license key.

To upgrade to 64-bit LispWorks, contact

```
lisp-sales@lispworks.com
```

7

Installation on UNIX

7.1 Introduction

This chapter is a brief installation guide for UNIX LispWorks 6.0. Chapter 11 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 11, “Configuration on UNIX”.

7.2 Extracting software from the CD-ROM

UNIX LispWorks 6.0 is supplied on a CD-ROM with the associated products CLIM 2.0, KnowledgeWorks, and LispWorks ORB. You will need root access while installing these products.

7.2.1 Finding out which CD-ROM files you need

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

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

Table 7.1 Platforms and associated codes

For HP PA (HP-UX 11x) you need the files named `lw60-hp-pa.tar` and `lwdoc60-unix.tar`.

For Sun Sparc (32-bit) you need the files named `lw60-sparc.tar` and `lwdoc60-unix.tar`.

For Sun Sparc (64-bit) you need the files named `lw60-sparc64.tar` and `lwdoc60-sparc64.tar`.

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

7.2.2 Unpacking the CD-ROM files

To unpack the CD-ROM files:

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

```
% tar -xof filename
```

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

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

Thus, an image named **lispworks-6-0-0-hp-pa11** would be a LispWorks 6.0 image for use on an HP PA machine running HP-UX 11.

7.3 Moving the LispWorks image and library

The LispWorks image must be able to find its library. The default library location is contained in the Lisp variable ***lispworks-directory***, but if that does not locate the library, LispWorks also can locate its library by a fallback mechanism which detects a numbered subdirectory **lib/6-0-0-0** alongside the image.

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

1. Put the LispWorks distribution in **/usr/lib/lispworks**. You will then have the LispWorks image at top-level in the **/usr/lib/lispworks** directory, and subdirectories **/usr/lib/lispworks/lib/6-0-0-0**.

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

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

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

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

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

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

See Section 7.5 below for more information about configuring LispWorks. You will need to install your license key first.

7.4 Obtaining and Installing your license keys

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

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

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

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

7.4.1.1 If you are using the keyfile system

You will need a valid key, placed in a keyfile, for LispWorks to run. Note that keys and licenses issued for use with LispWorks version 4.x do not work for LispWorks 6.0.

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

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

lisp-keys@lispworks.com

Other queries should be sent to

lisp-support@lispworks.com

although please be sure to check Section 12.9, “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 12.9.8, “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/6-0-0-0/config/keyfile.*hostname***, where ***hostname*** is the name of the host machine on which LispWorks is to run. The **lib** directory is expected by default to be located at **/usr/lib/lispworks/lib** (see Section 7.3 above)
- **lib/6-0-0-0/config/keyfile**, where the **lib** directory is as above.

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

For more details, see “How to obtain keys” on page 89.

7.4.1.2 If you are using the License Server

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

7.5 Configuring the LispWorks image

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

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

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

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

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

7.5.1 Saving a configured image

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

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

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

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

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

3. Start the supplied image using the configuration script as the build file. For example:

```
% lispworks-6-0-0-sparc-solaris -build /tmp/config.lisp
```

If the image will not run at this stage, it is probably not finding a valid key. See “Obtaining and Installing your license keys” on page 52

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.

7.5.2 Testing the newly saved image

The following steps provide a basic test of your installation.

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

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

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

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

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

7.6 Using the Documentation

Documentation in HTML and PDF formats is provided in a separate archive on the CD-ROM. If you want to access the documentation, you should unpack the appropriate archive named “Finding out which CD-ROM files you need” on page 50.

HTML documentation is installed in the **lib/6-0-0-0/manual/online** subdirectory of the LispWorks library, and can be accessed via the **Help** menu in the LispWorks IDE.

The PDF format manuals are installed in the **lib/6-0-0-0/manual/offline/pdf** subdirectory of the LispWorks library.

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

To use each of Delivery, LispWorks ORB, CLIM 2.0 and KnowledgeWorks you must obtain the required key and put in your keyfile. See “Keyfiles and the license server for HP PA and Sun Sparc (32-bit)” on page 52.

Then you need to load the layered product module. This is done by `(require "delivery")` or `(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 54.

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

8

Configuration on Mac OS X

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 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”

8.2 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/6-0-0-0/config` subdirectory of the LispWorks installation directory

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

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
- 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.

8.3.2 Configuring images for the different GUIs

If you have installed both the LispWorks images, for native Mac OS X and for GTK+, 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 GTK+ LispWorks image does not, and has **:gtk**.

8.3.3 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 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

It is not usually necessary to save an image merely to preload patches and your configuration, because these load very quickly on modern machines.

However, if you want to save an image to reduce startup time for a complex configuration (such as large application code) or to save a non-windowing image, then proceed as described in this section.

8.4.1 Create a configuration file

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made the desired changes in **my-configuration.lisp** you can save a new LispWorks image as described in “Create and use a save-image script” on page 60.

8.4.2 Create and use a save-image script

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

```
(in-package "CL-USER")
(load-all-patches)
(load "/tmp/my-configuration.lisp")
#+:cocoa
(save-image-with-bundle "/Applications/My LispWorks/LW")
#-:cocoa
(save-image "my-lispworks-gtk")
```

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

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

```
% cd "/Applications/LispWorks 6.0/LispWorks.app/Contents/MacOS"
```

or for the X11 / GTK+ LispWorks image:

```
% cd "/Applications/LispWorks 6.0"
```

3. Start the supplied image passing the configuration script the build file. For example enter one of the following commands (on one line of input):

```
% ./lispworks-6-0-0-macos-universal -build /tmp/save-config.lisp
```

or

```
% ./lispworks-6-0-0-macos-universal-gtk -build
/tmp/save-config.lisp
```

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

Saving the image takes some time.

You can now use the new **My LispWorks/LW.app** application bundle or the **my-lispworks-gtk** 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.

8.4.3 What to do if no image is saved

If no new image is saved, then there is some error while loading the build script. To see the error message, run the command with output redirected to a file, for example:

```
% ./lispworks-6-0-0-macos-universal -build /tmp/save-config.lisp
> /tmp/output.txt
```

Look in the file `/tmp/output.txt`.

8.4.4 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/GTK+ 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/GTK+ 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 and Reference Manual*, to further check that the configured image has been successfully built.
4. Test the load-on-demand system. In the Listener, type:

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

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

8.4.5 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 “Create and use a save-image script” 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. 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 6.0/LispWorks.app/Contents/MacOS/lispw
orks-6-0-0-macos-universal" -init my-lisp-init
```

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

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

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

```
% "/Applications/LispWorks 6.0/LispWorks.app/Contents/MacOS/lispw
orks-6-0-0-macos-universal" -init - -siteinit -
```

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

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

8.6 Loading CLIM 2.0

CLIM 2.0 is supported on the X11 / Motif GUI.

Load CLIM 2.0 into the "LispWorks for X11 IDE" 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 "/path/to/clim-lispworks")
```

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: CLIM is not supported under GTK+.

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")**.

8.7 The Common SQL interface

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

8.7.1 Loading Common SQL

To load Common SQL enter, for example:

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

or

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

Initialize the database type at runtime, for example:

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

or

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

See the *LispWorks User Guide and Reference Manual* for further information.

8.7.2 Supported databases

Common SQL on Mac OS X has been tested with DBMS Postgres 7.2.1, MySQL 5.0.18, Oracle Instant Client 10.2.0.4, ODBC driver PSQLODBC development code, and IODBC as supplied with Mac OS X.

8.7.3 Special considerations when using Common SQL

8.7.3.1 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.

8.7.3.2 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.

8.7.3.3 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.

8.7.3.4 Locating the Oracle, MySQL or PostgreSQL client libraries

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

8.8 Common Prolog and KnowledgeWorks

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

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

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

9

Configuration on Windows

9.1 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 3, “Installation on Windows”.

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

This chapter covers the following topics:

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

9.2 License keys

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

9.3 Configuring your LispWorks installation

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

9.3.1 Levels of configuration

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

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

In the first case, you use edited copies of files in the **`config`** 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 > Preferences...`** in the LispWorks IDE.

9.3.2 Configuration files available

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

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

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

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

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

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

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

config/a-dot-lispworks.lisp is a sample personal initialization file. You might like to copy this 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 9.4, below, and Section 9.5, “Initializing LispWorks” for further details.

9.4 Saving and testing the configured image

It is not usually necessary to save an image merely to preload patches and your configuration, because these load very quickly on modern machines.

However, if you want to save an image to reduce startup time for a complex configuration (such as large application code) or to save a non-windowing image, then proceed as described in this section.

9.4.1 Create a configuration file

Make a copy of **config\configure.lisp** called **C:\temp\my-configuration.lisp**. When you have made any desired changes in **my-configuration.lisp** you can save a new LispWorks image, as described in "Create and use a save-image script" on page 70.

9.4.2 Create and use a save-image script

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

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

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 the build file. For example:

   ```
   C:\Program Files\LispWorks>lispworks-6-0-0-x86-win32.exe -build
   C:\temp\save-config.lisp
   ```

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

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.

9.4.3 What to do if no image is saved

If the LispWorks splash screen appears briefly but no image is saved, then there is some error while loading the build script. To see the error message, run the command with output redirected to a file, for example:

```
C:\Program Files\LispWorks>lispworks-6-0-0-x86-win32.exe -build
C:\temp\save-config.lisp > c:\tmp\output.txt
```

Look in the file **c:\temp\output.txt**.

9.4.4 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 and Reference Manual*, to further check that the configured image has been successfully built.

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

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

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

9.4.5 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 “Create and use a save-image script” on page 70 but pass the **:environment** argument to **save-image**. For example:

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

9.5 Initializing LispWorks

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

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

```
C:\Program Files\LispWorks>lispworks-6-0-0-x86-win32.exe -init
my-lisp-init
```

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

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

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

```
C:\Program Files\LispWorks>lispworks-6-0-0-x86-win32.exe -init -
-siteinit -
```

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

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

9.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 6.0 with

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

and the CLIM demos with

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

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

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

```
(load-all-patches)
(require "clim")
(save-image "C:\\path\\to\\clim-lispworks")
```

9.6.1 Running the CLIM demos

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

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

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

9.7 The Common SQL interface

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

9.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

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

and at runtime call:

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

and then you can connect to any installed ODBC datasource.

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

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

and at runtime call:

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

See the *LispWorks User Guide and Reference Manual* for further information.

9.8 Common Prolog and KnowledgeWorks

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

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

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

10

Configuration on Linux, x86/x64 Solaris, and FreeBSD

10.1 Introduction

This chapter explains how to get your LispWorks Professional or Enterprise Edition up and running on Linux, x86/X64 Solaris, or FreeBSD, having already installed the files from the CD-ROM into an appropriate directory. If you have not done this, refer to Chapter 4, Installation on Linux, Chapter 5, Installation on x86/x64 Solaris or Chapter 6, Installation on FreeBSD.

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

This chapter covers the following topics:

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

- "Common Prolog and KnowledgeWorks"

10.2 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/6-0-0-0/config` subdirectory of the LispWorks installation directory

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

10.3 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.

10.3.1 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 > Preferences...** in the LispWorks IDE.

10.3.2 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 10.4, below, and Section 10.5, “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 10.4, below, and Section 10.5, "Initializing LispWorks" for further details.

10.4 Saving and testing the configured image

It is not usually necessary to save an image merely to preload patches and your configuration, because these load very quickly on modern machines.

However, if you want to save an image to reduce startup time for a complex configuration (such as large application code) or to save a non-windowing image, then proceed as described in this section.

10.4.1 Create a configuration file

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made any desired changes in **my-configuration.lisp** you can save a new LispWorks image, as described in "Create and use a save-image script" on page 78.

10.4.2 Create and use a save-image script

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

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

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 the build file. For example:

   ```
   % lispworks-6-0-0-x86-linux -build /tmp/save-config.lisp
   ```

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

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.

10.4.3 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 and Reference Manual*, to further check that the configured image has been successfully built.
4. Test the **load-on-demand** system. In the Listener, type:

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

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

10.4.4 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 “Create and use a save-image script” on page 78 but pass the **:environment** argument to **save-image**. For example:

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

10.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. ~ 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-6-0-0-x86-linux -init my-lisp-init
```

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

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

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

```
% lispworks-6-0-0-x86-linux -init - -siteinit -
```

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

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

10.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 6.0 with

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

and the CLIM demos with

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

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

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

```
(load-all-patches)
(require "clim")
(save-image "/path/to/clim-lispworks")
```

10.6.1 Running the CLIM demos

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

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

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

10.7 The Common SQL interface

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

10.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

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

and at runtime call:

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

and then you can connect to any installed ODBC datasource.

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

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

and at runtime call:

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

See the *LispWorks User Guide and Reference Manual* for further information.

10.8 Common Prolog and KnowledgeWorks

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

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

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

10.9 Documentation on x86/x86 Solaris and FreeBSD

Except where explicitly mentioned, information stated as specific to LispWorks for Linux also applies to LispWorks for x86/x64 Solaris and LispWorks for FreeBSD.

11

Configuration on UNIX

11.1 Disk requirements

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

Installing the documentation adds up to 66MB to this. You can delete some of these files if you wish, for example you might not need the PDF manuals in `lib/6-0-0-0/manual/offline/pdf` (28Mb). You can download these PDF format manuals from `www.lispworks.com/documentation` at any time, and the same manuals are also available there in PostScript format. However, note that the **Help** menu commands will not work if you corrupt the `lib/6-0-0-0/manual/online` directory of the LispWorks library.

11.2 Software Requirements

The LispWorks 6.0 for UNIX GUI requires X11 release 5 or above, Motif version 2 and Imlib.

Imlib version 1.9.13 or later is recommended. Lisp Support can supply a suitable Imlib library.

11.3 The CD-ROM

This section explains the organization of the LispWorks 6.0 CD-ROM which contains the LispWorks products you have bought, and how to mount it.

11.3.1 The LispWorks 6.0 CD-ROM

The CD-ROM contains images for LispWorks 6.0 and associated products on your platform or platforms.

11.3.1.1 CD-ROM format

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

11.3.2 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. This is described in “Mounting the CD-ROM” on page 84.
2. Extract the product files from the `tar` file containing them. This is described in “Installing LispWorks” on page 85.

11.3.3 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.

11.3.3.1 HP UX (HP Precision Architecture)

To mount:

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

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

To unmount:

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

Again, use the appropriate device designation for your hardware.

11.3.3.2 Solaris (Sun Sparc)

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

```
/cdrom/lw_60/
```

To unmount:

```
umount /cdrom/lw_60/
```

11.4 Installing LispWorks

This section explains how to install LispWorks, having already mounted the CD-ROM. If you have not done this, refer to Section 11.3, “The CD-ROM”. It also describes how you obtain keys to run LispWorks 6.0.

11.4.1 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.

There are subsections below explaining the process for each supported platform.

11.4.1.1 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.

11.4.1.2 Keeping your old LispWorks installation

You can install LispWorks 6.0 in the same directory as previous versions of LispWorks such as LispWorks 5.1. This is because all the 6.0 files are stored in a subdirectory called `lib/6-0-0-0`.

You must recompile all your code with the LispWorks 6.0 compiler.

Binaries produced by `compile-file` in previous versions of LispWorks such as LispWorks 5.1 do not load into a LispWorks 6.0 image.

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

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

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

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

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

For example, to extract the files for LispWorks (32-bit) on SPARC Solaris, with the CD-ROM mounted at **/cdrom/lw_60/**, you would type

```
tar -xof /cdrom/lw_60/lw60-sparc.tar
```

11.4.1.4 HP UX (HP Precision Architecture)

The files you need to unpack for LispWorks on HP UX are **lw60-hp-pa.tar** and **lwdoc60-unix.tar**.

The LispWorks image is

```
./lispworks-6-0-0-hp-pa11
```

11.4.1.5 SPARC Solaris (LispWorks 32-bit)

The files you need to unpack for LispWorks (32-bit) on Solaris are **lw60-sparc.tar** and **lwdoc60-unix.tar**.

The LispWorks image is

```
./lispworks-6-0-0-sparc-solaris
```

11.4.1.6 SPARC Solaris (LispWorks 64-bit)

The files you need to unpack for LispWorks (64-bit) on Solaris are **lw60-sparc64.tar** and **lwdoc60-sparc64.tar**.

The LispWorks image is

```
./lispworks-6-0-0-sparc64-solaris
```

11.4.2 Keyfiles and how to obtain them

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

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

11.4.2.1 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/6-0-0-0/config** (see "If you are using the keyfile system" on page 52.
- **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/6-0-0-0/config**.

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

11.4.2.2 The contents of a keyfile

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

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

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

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

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

11.4.2.3 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 108.

Once you have the key, write it into a file in one of the places listed in Section 11.4.2.1, and start up the LispWorks image.

11.4.3 The License Server

This section applies only to HP PA and Sun Sparc (32-bit). There is no license server for LispWorks (64-bit) for Solaris.

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.

11.5 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.

11.5.1 The LispWorks image

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

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

Thus, an image named **`lispworks-6-0-0-sparc-solaris`** is the LispWorks 6.0 image for use on Sun Sparc (32-bit) Solaris machines.

There may be several images on the CD-ROM, one for each of the architectures LispWorks can run on.

As noted in Section 11.4.1.1 on page 86, once installed, the basic executable Lisp image can be placed somewhere in the UNIX file system likely to be on its users' search path. A suitable place might be **`/usr/local/bin/lispworks`**.

11.5.2 The LispWorks library

The runtime directory structure (basically, everything except the image file) should be somewhere publicly readable: **`/usr/lib/lispworks`**, by default. If there is not enough room in any of the normal publicly accessible locations, you could put a symbolic link there pointing to the installation directory in a partition with more disk space. The installation directory must contain a subdirectory called **`lib/6-0-0-0/`**.

Among the directories on this subdirectory are the following:

- **`config`** — various files that can be adjusted in order to customize the image (see Section 11.7 on page 92).
- **`app-defaults`** — X / Motif resources for LispWorks and the Lisp Monitor.
- **`postscript`** — printer descriptions for the CAPI printing interface.

- **etc** — the executable for the Lisp Monitor.
- **load-on-demand** — Lisp library code that is loaded into a running LispWorks system as and when required.
- **patches** — numbered patches to LispWorks and layered products.
- **private-patches** — the location to place private (named) patches that Lisp support may send to you.
- **examples** — directories containing various code examples, including most of the code printed in the user documentation.
- **translations** — the place for logical pathname translations settings
- **src** — source code supplied with LispWorks

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

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

By default, all these directories are assumed to reside beneath **/usr/lib/lispworks/lib/6-0-0-0/**, although you may place the **lib** directory somewhere else.

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

11.6 Printing copies of the LispWorks documentation

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

The **offline/pdf** directory contains each manual in PDF format.

11.7 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.)

11.7.1 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.

11.7.2 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 11.7.3 below, and see also Section 11.8, “LispWorks initialization arguments” for further details.

11.7.3 Saving and testing the configured image

It is not usually necessary to save an image merely to preload patches and your configuration, because these load very quickly on modern machines.

However, if you want to save an image to reduce startup time for a complex configuration (such as large application code) or to save a windowing image, then proceed as described in this section.

11.7.4 Create a configuration file

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made any desired changes in **my-configuration.lisp** you can save a new LispWorks image, as described in "Create and use a save-image script" on page 94.

11.7.5 Create and use a save-image script

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

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

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

   ```
   unix% lispworks-6-0-0-sparc-solaris -init - -siteinit -
   ```

   If the image will not run at this stage, it is probably not finding a valid key. See "Keyfiles and how to obtain them" on page 87.

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

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

   Now load all current patches:

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

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

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

Saving the image takes some time.

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

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

11.7.5.1 Testing the newly saved image

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

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

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

   Before information about the fixnum 1 is printed, the system should load the inspector from the **load-on-demand** directory.
4. Next, test the ability of the system to interface to a local X server. If necessary, start an X server either on the local machine or on a machine networked to it. Type:

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

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

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

11.8 LispWorks initialization arguments

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.

12

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.

12.1 Troubleshooting

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

12.1.1 License key errors in the Professional and Enterprise Editions

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

These are the possible problems:

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

On Linux, x86/x64 Solaris and FreeBSD, this is also a possible cause of the problem:

- The machine name has changed since LispWorks was installed.

On Mac OS X, Linux, x86/x64 Solaris and FreeBSD, the key is expected to be stored in a keyfile, and an appropriate error message is printed at the terminal for each case. If this message does not help you to resolve the problem, report it to Lisp Support and include the terminal output.

On Windows, the key is expected to be stored in the Windows registry. If you cannot resolve the problem, export your HKEY_LOCAL_MACHINE\SOFTWARE\LispWorks registry tree and include this with your report to Lisp Support.

12.1.2 Failure of the load-on-demand system

Module files are in the modules directory **`lib/6-0-0-0/load-on-demand`** under **`*lispworks-directory*`**.

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

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

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

12.1.3 Memory requirements

To run the full LispWorks system, with its GUI, you will need around 30MB of swap space for the image 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.

Check the size of the image, both by **cl:room** and by OS facilities (such ps or top on *nix, Task Manager on Windows) to see if all the sizes are as expected. If there are large discrepencies, check them.

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. In cases where external libraries are mapped above LispWorks and inhibit its growth, you may be able to relocate LispWorks, as described under "Startup relocation" in the *LispWorks User Guide and Reference Manual*.

12.2 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 12.7, "Troubleshooting on X11 / Motif" below for issues specific to X11 / Motif.

12.2.1 Default installation requires administrator on Mac OS X

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

However, a non-administrator may install LispWorks elsewhere.

12.2.2 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 for Editor and Listener tools by **LispWorks > Preferences... > Environment > Styles > Editor Font**.

12.3 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 103 below for issues specific to X11 / Motif.

12.3.1 Processes hanging

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

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

LispWorks 6.0 supports kernel versions 2.4.20 and later.

12.3.2 RPM_INSTALL_PREFIX not set

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

```
# rpm --install tmp/lispworks-clim-6.0-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-6.0-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-6.0-1
#
```

This is only a problem when LispWorks itself was installed in a non-default location (that is, using the `--prefix` RPM option). You would then want to

supply that same **--prefix** value when installing the secondary rpm. A bug in RPM means that a required environment variable **RPM_INSTALL_PREFIX** is not set automically to the supplied value. We have seen this bug in RPM version 4.2, as distributed with RedHat 8 and 9.

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

```
rpm --install --prefix /usr/lisp lispworks-6.0-1.i386.rpm
```

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

```
setenv RPM_INSTALL_PREFIX /usr/lisp
rpm --install --prefix /usr/lisp lispworks-clim-6.0-1.i386.rpm
```

12.3.3 Using multiple versions of Motif on Linux

The version of Open Motif required by LispWorks 6.0 with the Motif GUI may not be compatible with other applications (including LispWorks 4.2). It is however compatible with LispWorks 5.1, LispWorks 5.0, LispWorks 4.4 and 4.3, so you for example you should be able to run LispWorks 6.0, LispWorks 5.1 and LispWorks 4.4 simultaneously with either Open Motif installed.

Whilst it is not supported for LispWorks 5.1, you can still use LessTif for LispWorks 5.0 and earlier - see the Installation Guide for that version for details.

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 has been installed in a non-standard location, you can set the environment variable **LD_LIBRARY_PATH** to allow an application to find that library. Specifically, if <*motiflibdir*> denotes the directory containing the Motif 2.2 file **libXm.so** then set **LD_LIBRARY_PATH** to include <*motiflibdir*>.

Note: to find out which version of libXm your LispWorks 6.0 image is actually using, look in the bug form. See “Generate a bug report template” on page 109 for instructions on generating the bug form.

12.4 Troubleshooting on x86/x64 Solaris

This section describes some of the most common problems that can occur during installation or configuration of LispWorks for x86/x64 Solaris.

See also "Problems with CAPI on GTK+" on page 153 and "Troubleshooting on X11/Motif" on page 103.

12.4.1 GTK+ version

GTK+ version 2.4 or higher is required to run the LispWorks image as distributed.

12.5 Troubleshooting on FreeBSD

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

See also "Troubleshooting on X11/Motif" on page 103 below for issues specific to X11/Motif.

12.5.1 Poor latency when using multiple threads

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

12.6 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, x86/x64 Solaris or FreeBSD).

See also "Troubleshooting on X11/Motif" on page 103 for issues specific to X11/Motif.

12.6.1 Problems with CD-ROM file system

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

```
lwdoc60-unix.tar;1
```

then check the **mount** command used (“Mounting the CD-ROM” on page 84).

12.6.2 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/6-0-0-0**) or be executed in the same directory as the keyfile. If the image has been configured, check that the keyfile is in the right place and that the value of ***lispworks-directory*** is correct.

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

12.7 Troubleshooting on X11/Motif

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

12.7.1 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.

12.7.2 Problems with fonts on Motif

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/Motif resources" on page 105 for more information.

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

12.7.3 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/Motif resources" on page 105 for more information.

To remove the problem, you can then change the resources (for example, by editing the file mentioned in "X11/Motif resources" on page 105) to reduce the number of colors LispWorks allocates.

12.7.4 Motif 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.)

12.7.5 Non-standard X11/Motif key bindings

On 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 12.7.6):

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

12.7.6 X11/Motif resources

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

12.7.7 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 6.0/lispworks-6-0-0-macos-
universal-motif can't open library:
/usr/local/lib/libXm.4.dylib (No such file or directory, errno
= 2)
.
```

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

12.8 Updating with patches

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

12.8.1 Extracting simple patches

Save the email attachment to your disk.

See Section 12.8.3.2, “Private patches” below about location of your private patches.

12.8.2 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.

12.8.3 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.

12.8.3.1 Public patches

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

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

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

You should consider saving a new image with the latest patches pre-loaded, as described in Section 8.4, “Saving and testing the configured image” (Mac OS X), Section 9.4, “Saving and testing the configured image” (Windows) or Section 10.4, “Saving and testing the configured image” (Linux, x86/x64 Solaris or FreeBSD), or Section 11.7.3, “Saving and testing the configured image” (other UNIX).

12.8.3.2 Private patches

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

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

<*install*>/LispWorks 6.0/Library/lib/6-0-0-0/private-patches/my-patch.nfasl

On Windows:

<*install*>\lib\6-0-0-0\private-patches\my-patch.ofasl

On Linux:

<*install*>/lib/6-0-0-0/private-patches/my-patch.ufasl

On UNIX:

<*install*>/lib/6-0-0-0/private-patches/my-patch.pfasl (for HP-PA)
<*install*>/lib/6-0-0-0/private-patches/my-patch.wfasl (for SPARC)

You will receive a Lisp form needed to load such a patch, such as

```
(LOAD-ONE-PRIVATE-PATCH "my-patch" :SYSTEM)
```

This form should be added to the **flet** form 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 8.4, “Saving and testing the configured image” (Mac OS X), Section 9.4, “Saving and testing the configured image” (Windows), Section 10.4, “Saving and testing the configured image” (Linux, x86/x64 Solaris or FreeBSD), or Section 11.7.3, “Saving and testing the configured image” (other 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.

12.9 Reporting bugs

If you discover a bug, in either the software or the documentation, you can submit a bug report by any of the following routes.

- email
- fax
- paper mail (post)
- telephone

The addresses are listed in Section 12.9.8. Please note that we much prefer email.

12.9.1 Check for existing fixes

Before reporting a bug, please ensure that you have the latest patches installed and loaded. Visit **www.lispworks.com/downloads/patch-selection.html** for the latest patch release.

If the bug persists, check the Lisp Knowledgebase at **www.lispworks.com/support/** for information about the problem - we may already have fixed it or found workarounds.

If you need informal advice or tips, try joining the LispWorks users' mailing list. Details are at **www.lispworks.com/support/lisp-hug.html**.

12.9.2 Performance Issues

If the problem is poor performance, you should use **room**, **extended-time** and **profile** to check what actually happens. See the *LispWorks User Guide and Reference Manual* for details of these diagnostic functions and macros.

If this does not help you to resolve the problem, submit a report to Lisp Support (see next section) and attach the output of the diagnostics.

12.9.3 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.

12.9.4 Add details to your bug report

Under 'Urgency' tell us how urgent the issue is for you. We classify reports as follows:

ASAP	A bug or missing feature that is stopping progress. Probably needs a private patch, possibly under a support contract, unless a workaround can be found.
Current Release	Either a fix in the next patch bundle or as a private patch, possibly under a support contract.
Next Release	A fix would be nice in the next minor release.
Future Release	An item for our wishlist.
None	Probably not a bug or feature request.

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 (> 2MB) 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 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.

12.9.5 Reporting crashes

Very occasionally, there are circumstances where it is not possible to generate a bug report form from the running Lisp which has the bug. For example, a delivered image may lack the debugger, or the bug may cause lisp to crash completely. In such circumstances:

1. It is still useful for us to see a bug report form from your lisp image so that we can see your system details. Generate the form before your code is loaded or a broken call is made, and attach it to your report.
2. Create a file **init.lisp** which loads your code that leads to the crash.
3. Run LispWorks with **init.lisp** as the initialization file and with output redirected to a file. For example, on Mac OS X:

   ```
   % "/Applications/LispWorks 6.0/LispWorks.app/Contents/MacOS/lispw
   orks-6-0-0-macos-universal" -init init.lisp > lw.out
   ```

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

   On Windows:

```
C:\> "Program Files\LispWorks\lispworks-6-0-0-x86-
win32.exe" -init init.lisp > lw.out
```

where `C:\>` denotes the prompt in a MS-DOS command window.

On Linux:

```
% /usr/bin/lispworks-6-0-0-x86-linux -init init.lisp > lw.out
```

where `%` denotes a Unix shell prompt.

On UNIX (SPARC in this example):

```
% /usr/lib/lispworks/lib/6-0-0-0/config/lispworks-6-0-0-sparc-
solaris -init init.lisp > lw.out
```

4. Attach the `lw.out` file to your report. In general it is not useful to edit the output of your Lisp image, so please send it as-is. Where output files are very large (> 2MB) and repetitive, the first and last 200 lines might be adequate.

12.9.6 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.

You can make the application write a bug form to a log file automatically by making your error handlers call `dbg:log-bug-form`.

12.9.7 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.)

12.9.8 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 109 for details. You can always get a backtrace from within the debugger by entering `:bb` at the debugger prompt

12.9.9 Sending large files

Note: Please check with Lisp Support in advance if you are intending to send very large (> 2MB) files via email.

12.9.10 Information for Personal Edition users

We appreciate feedback from users of LispWorks Personal Edition, and often we are able to provide advice or workarounds if you run into problems. However please bear in mind that this free product is unsupported. For informal

advice and tips, try joining the LispWorks users mailing list. Details are at `www.lispworks.com/support/lisp-hug.html`.

12.10 Transferring LispWorks to a different machine

This section lists the steps necessary to transfer your LispWorks Professional or Enterprise Edition license to another machine.

1. Install LispWorks on your new machine.
2. Add latest patch bundle.
3. If you received private patches (named patch files, in the `lib/6-0-0-0/private-patches` directory) for this version of LispWorks, move them and your `private-patches/load.lisp` file to the corresponding location in the new installation.
4. Test the new installation by running LispWorks and check the patch banner in the output of **Help > Report Bug**. It should be identical to the original installation, If it is not, check that the public patches have been installed, and that you private patches have been moved to the new `private-patches` folder, along with the `load.lisp` file.

Please note that the LispWorks EULA restricts multiple installations so you need to remove the original installation. Instructions for uninstalling LispWorks are in the per-platform chapters of this manual.

Some operating systems provide ways to copy software to another machine. A copied LispWorks installation will not run. Please contact Lisp Support if you want to install your license to a copied installation of LispWorks.

13

Release Notes

13.1 Platform support

13.1.1 x86/x64 Solaris

LispWorks (32-bit) for x86/x64 Solaris and LispWorks (64-bit) for x86/x64 Solaris are now available.

13.1.2 Running on 64-bit machines

As far as we know each of the 32-bit LispWorks implementations runs correctly in the 32-bit subsystem of the corresponding 64-bit platform.

13.1.3 Older platforms

LispWorks 6.0 does not support Windows Millennium Edition, Windows 2000, FreeBSD 5 or RedHat 9.

LispWorks 6.0 for Macintosh supports MacOS X 10.3, 10.4 and 10.5 on PowerPC, and on MacOS X 10.4, 10.5 and 10.6 on Intel.

13.2 Symmetric Multiprocessing

LispWorks now supports SMP on Microsoft Windows, Mac OS X, Linux, FreeBSD and x86/x64 Solaris platforms.

Some new functionality is implemented only on these platforms. Where functionality differs from other platforms, the documentation refers to "SMP LispWorks" or "Non-SMP LispWorks", as appropriate.

13.2.1 Old interrupt blocking APIs removed

The macros **`mp:without-interrupts`** and **`mp:without-preemption`**, which were available in LispWorks 5.1 and earlier, are no longer supported. The semantics of these macros allowed them to be used for several different purposes, which now require specific solutions.

Code which must be run to completion without being interrupted should use the new interrupt-blocking functionality. See “New ways to block interrupts” on page 116.

Code which needs to be atomic should be converted to use locks (see the "Multiprocessing" chapter of the *LispWorks User Guide and Reference Manual*) or the new low level atomic operations (see “Atomic operations” on page 116).

13.2.2 New ways to block interrupts

Where a set of operations must be completed without being interrupted by **`mp:process-interrupt`**, keyboard breaks and so on, you should use the new interrupt-blocking functionality. See "Blocking interrupts" in the *LispWorks User Guide and Reference Manual* for details.

13.2.3 Atomic operations

New operations are provided to perform atomic operations on some Common Lisp places. See "Low level atomic operations" in the *LispWorks User Guide and Reference Manual* for details.

13.2.4 New features of locks

mp:make-lock now offers control over whether a lock can be locked recursively. Non-recursive locks can be useful for debugging code where a lock is not expected to be claimed recursively.

mp:make-lock also allows you to create "sharing" locks which support sharing and exclusive locking.

See **mp:make-lock** in the *LispWorks User Guide and Reference Manual* for details.

There are new APIs for querying locks: **mp:lock-recursive-p**, **mp:lock-owned-by-current-process-p**, **mp:lock-locked-p** and **mp:lock-recursively-locked-p**.

Additionally **mp:lock-owner** now checks for an exclusive lock in the case of a sharing lock.

The performance of locks has been improved.

13.2.5 Efficient ways to synchronize between threads

In LispWorks 5.1 and previous versions, the main way to synchronize between threads is to use **mp:process-wait** or **mp:process-wait-with-timeout** to supply a predicate to the scheduler. The predicate runs periodically in the background to identify threads that are no longer blocked.

These functions are still available, but there are some alternatives that can be more efficient in many cases by removing the need for the scheduler. The alternatives are:

- Mailboxes (FIFO queues)
- Condition Variables (used with a lock)
- Barriers (counting arrivals at a certain point in the code)
- Counting Semaphores (limiting the number of users of a shared resource)

The new synchronization objects are described in "Synchronization between threads" in the *LispWorks User Guide and Reference Manual*. There are also new "local" variants of **mp:process-wait** which reduce the overhead on the scheduler. See "Process Waiting" in the *LispWorks User Guide and Reference Manual*.

The new function **mp:current-process-pause** is like **cl:sleep**, except that it allows another process to wake up the current process.

The new function **mp:process-join** provides a way to wait until a process dies which returns without any delay after the process dies.

Lastly, there are new synchronization functions which can ensure the order of store and load operations across multiple threads. See "Ensuring order of memory between operations in different threads" in the *LispWorks User Guide and Reference Manual*.

13.2.6 Access to specials

In SMP LispWorks, access to non-constant special variables is a little slower than in previous releases. You can speed up such access by declarations of the special symbol, normally by using **cl:proclaim** or **cl:declaim** with one of the new declarations **hcl:special-global**, **hcl:special-dynamic** and **hcl:special-fast-access**. See "Optimizing your code" in the *LispWorks User Guide and Reference Manual*.

13.2.7 Processing and handling events

mp:process-all-events processes the events in the mailbox (the event queue) of the current process.

If you use your own classes to represent events in the mailbox of a process, you can add specific handling via **mp:general-handle-event**.

13.2.8 Automatic creation of a mailbox

mp:process-run-function now allows you to specify that it creates a mailbox and associates it with the process. You do not need to explicitly call **mp:make-mailbox** and **(setf mp:process-mailbox)** as in previous versions of LispWorks.

13.2.9 Sending and interpreting events

mp:process-send now sends any Lisp object as the event. It is the responsibility of the receiving process to read and interpret this event object.

Also, **mp:process-send** now ensures there is a mailbox in the receiving process.

13.2.10 mp:mailbox-read distinguishes a read value from a timeout

mp:mailbox-read now returns a second value which is true when a value was actually read, but **nil** if **mp:mailbox-read** timed out instead.

13.2.11 mp:process-plist deprecated

mp:process-plist is deprecated, because it does not work well in SMP LispWorks. Processes now have general properties and private properties; use one of these instead. See "Process properties" in the *LispWorks User Guide and Reference Manual* for details.

13.2.12 mp:process-event-queue deprecated

mp:process-event-queue is deprecated, because it does not check its arguments. Instead, use **mp:process-mailbox** which accesses the mailbox (event queue) of a process safely.

13.3 GTK+ window system

LispWorks now uses GTK+ as the default window system for CAPI and the LispWorks IDE on Linux, FreeBSD and x86/x64 Solaris. GTK+ is also supported on Mac OS X as an alternative to Cocoa. LispWorks requires GTK+ version 2.4 or higher.

Note: LispWorks on SPARC Solaris and HP-UX does not support GTK+.

A few known problems are documented on “Problems with CAPI on GTK+” on page 153.

13.3.1 Using Motif instead of GTK+

Use of Motif with LispWorks on Linux, FreeBSD, x86/x64 Solaris and Mac OS X is deprecated, but it is available by

```
(require "capi-motif")
```

To use LispWorks 6.0 with Motif you also need Imlib installed, as described earlier in this manual.

13.4 New CAPI features

See the *LispWorks CAPI Reference Manual* for more details of these.

13.4.1 Break gesture available in CAPI Cocoa applications

On Cocoa the gesture **Command+Ctrl+,** (comma) is now the break gesture.

When the break gesture is used, LispWorks attempts to break the current operation in a useful way, allowing the user to regain control even if the GUI is busy. This is useful if the "spinning beach ball" cursor appears for some time and the application appears to be hanging.

13.4.2 Setting the break gestures

On GTK+ and Motif the new function **capi:set-interactive-break-gestures** sets the gestures that can be used to break by typing at an interface.

13.4.3 Lisp mode parenthesis coloring

The function **capi:set-editor-parenthesis-colors** sets the colors that are used for the new parenthesis coloring in a **capi:editor-pane** in Lisp mode.

13.4.4 Toolbar API for capi:interface

There is a new CAPI facility for making a toolbar, which is typically displayed at the top of the window. On Cocoa, this makes a standard foldable toolbar. The user can use a customization dialog (via the context menu) to choose which items appear on the toolbar. See the *toolbar-items*, *toolbar-states* and *default-toolbar-states* initargs for **capi:interface** and the functions **capi:interface-toolbar-state**, **capi:interface-default-toolbar-states**, **capi:interface-update-toolbar** and **capi:interface-customize-toolbar**.

13.4.5 Lightweight positioning of panes

The new class **capi:static-layout** allows you to position panes inside it without the overhead of **capi:pinboard-layout** (which is more expensive because it is an output pane).

Existing code that uses **capi:pinboard-layout** still works.

13.4.6 Grid cells spanning multiple columns or rows

capi:grid-layout now supports cells spanning multiple columns or rows. See the new keywords **:right-extend** and **:bottom-extend** and the example in **examples/capi/layouts/extend.lisp**.

13.4.7 Separators in rows and columns

capi:row-layout and **capi:column-layout** now support separators. These are like dividers (supported in earlier versions) but are not movable by the user. To add a separator, include **:separator** in the *description* of the layout.

13.4.8 Ratio specification for dividers and separators

If you specify *ratios* for a **capi:row-layout** or **capi:column-layout** which has **:divider** in its *description*, the ratio for the divider should be **nil**, since the divider is always displayed at its minimum size in the relevant dimension.

This requirement also applies to the new **:separator**.

In LispWorks 5.1 and previous versions, a numerical value of the ratio for the divider is accepted.

13.4.9 Lists with a filter

capi:list-panel now supports adding a filter above the list. When the user has entered something in the filter only matching items are displayed, and **capi:collection-items** returns only the displayed items, and **capi:choice-selection** indexes into the displayed items.

There are new accessors for the unfiltered items.

Various new initargs to **capi:list-panel** allow control over the filter appearance and behavior of the filter.

13.4.10 Lists which do not take input focus

The functions **capi:prompt-with-list-non-focus** and dispalys a list in a non-focus window, which does not take the input focus and hence does not see keyboard input in general. It responds to mouse gestures. Other APIs allow you to specify particular keyboard gestures that are seen, control a filter on the list, close the non-focus window and so on.

This mechanism is used in the LispWorks IDE in various commands which offer completion of input.

capi:text-input-pane-in-place-complete uses this mechanism to provide in-place completion in a text input pane.

13.4.11 Finding the child pane with input focus

The new function **capi:pane-descendant-child-with-focus** returns the child pane of a specified parent pane or layout that currently has the input focus, if there is one.

13.4.12 Automatic scrolling to show the focus pane

The new *scroll-if-not-visible-p* attribute of a **capi:simple-pane** and subclasses controls scrolling behavior of the parent when the pane is given the input focus. You can specify that it scrolls to make the pane fully visible (where possible) always, never or only when the input focus was set by a non-mouse gesture. For more details see the accessor **capi:scroll-if-not-visible-p**.

13.4.13 Programmatic scrolling of lists etc implemented on Cocoa

The functions **capi:scroll** and **capi:get-scroll-position** can now be used on Cocoa to programmatically scroll **capi:list-panel** and other panes which have scroll bars.

This functionality is not implemented in LispWorks 5.1 for Macintosh and earlier versions.

13.4.14 Edit/select operations for the active pane

A new interface allows you to invoke edit and selection operations on the active pane (the pane with input focus) within an interface. For example **capi:active-pane-copy** finds the active pane and tells it to do a copy operation, which is implemented by the generic function **capi:pane-interface-copy-object**. You can supply methods for your own pane and interface classes.

Copy, cut, paste, select, deselect and undo operations and associated predicates can be invoked in the same way.

13.4.15 Buffered output to improve drawing on Windows and Motif

If redrawing of your **capi:pinboard-layout** or **capi:output-pane** flickers on Microsoft Windows or Motif, perhaps because there are many pinboard objects or the redisplay is complex in some other way, you can use the new initarg **:draw-with-buffer**. This makes a pixmap to buffer the output before drawing it to the screen.

The **capi:output-pane** initarg **:draw-with-buffer** supersedes the **capi:pinboard-layout** initarg **:draw-pinboard-objects** (which was not documented but was used in some CAPI examples distributed with LispWorks 5.1 and earlier versions). Use **:draw-with-buffer** instead.

:draw-with-buffer has no effect on Cocoa and GTK+ because these libraries always buffer the output.

13.4.16 Detecting key presses and releases

CAPI now supports a portable way to detect key press and release gestures, by using **:key** in the *input-model*. This is different from using **:character** or **:gesture-spec**, because it captures the key and modifiers pressed or released, not the character that they would generate. It should only be used where you require low level access to the keyboard input. See the description of *input-model* in **capi:output-pane** for the details.

13.4.17 Modifiers in button and motion input-model gestures

All modifier keys (**:shift**, **:control**, **:meta** and **:hyper**) now work in button and motion *input-model* gestures. See the description of *input-model* in **capi:output-pane** for the details.

13.4.18 Stop playing of a sound file

The new function **capi:stop-sound** stops a sound from playing. You could use this to interrupt playing a MIDI file.

13.4.19 Drag'n'drop enhancements

This section describes enhancements to the drag and drop functionality which was introduced in LispWorks 5.1.

13.4.19.1 Drag'n'drop in lists and trees

capi:list-panel and **capi:tree-view** can now allow items to be dragged for use with the drag and drop APIs. See the new initarg **:drag-callback** for **capi:simple-pane**.

Additionally, on Cocoa **capi:list-panel** and **capi:tree-view** can now accept objects dropped using the drag and drop APIs. See the initarg **:drop-callback** for **capi:simple-pane**.

13.4.19.2 Drag lists of files on Cocoa

Dragging a list of files to LispWorks or another application such as the Finder now works. Call **capi:drag-pane-object** with **:filename-list** in the *plist*.

13.4.19.3 Drop coordinates on Microsoft Windows

On Windows, when doing drag and drop operations in a scrollable output pane the coordinates (**capi:drop-object-pane-x** and **capi:drop-object-pane-y**) now include the current **capi:%scroll-x%** and **capi:%scroll-y%** geometry slot values. This now matches the behavior on Cocoa.

13.4.20 Matching GTK+ and Motif resources

You can configure your application to use resources on GTK+ and X11 / Motif. (This was already true on Motif, but is now documented.)

An example showing how to make a CAPI GUI configurable by GTK+ resources is in **examples/capi/elements/gtk-resources.lisp**.

Example resource files are in **examples/gtk/**.

For more information see the sections "Matching resources" in the *LispWorks CAPI User Guide* and the description of *widget-name* for **capi:element** in the *LispWorks CAPI Reference Manual*.

13.4.21 Callback types including element

Subclasses of **capi:callbacks** now support new values of *callback-type* **:element**, **:element-item** and so on. These allow you to specify that the callback is called on the element (that is, a pane or menu item) that contains it.

13.4.22 read-only editor-panes

You can now specify *enabled* **:read-only** to make a **capi:editor-pane** be read-only, which means that input to the pane by keyboard or mouse gestures cannot change the text and the **Cut** and **Paste** menu entries are disabled. Programmatic modifications of the text are still allowed.

13.4.23 Uniform error handling in dialogs

capi:popup-confirmer and **capi:display-dialog** have a new keyword argument **:callback-error-handler**. This keyword allows uniform error handling in callbacks across platforms.

The new function **capi:current-popup** allows you to find the dialog itself in the scope of the callbacks.

13.4.24 Dialogs that are dismissed by any input

The new value **:dismiss-on-input** for the *modal* argument to **capi:popup-confirmer** and **capi:display-dialog** specifies that the dialog is application-

modal but any user gesture (a button or key press) causes the dialog to disappear.

13.4.25 Enabling menu items when a dialog is on screen

On Cocoa, **capi:menu-item** now supports enabling its items even when a dialog is on the screen, via the new initarg **:enabled-function-for-dialog**.

This does not apply on other platforms, since the menu bar is not accessible when a modal dialog is on the screen.

13.4.26 Support for file packages as directories on Cocoa

The new keyword argument *file-package-is-directory* to **capi:prompt-for-file** allows you to specify that a file package should be treated as a directory.

file-package-is-directory corresponds to the **treatsFilePackagesAsDirectories** method of **NSSavePanel** in Cocoa. It has no effect on other platforms.

13.4.27 Efficient modification of a range-pane

The new function **capi:range-set-sizes** set the *start*, *end*, *slug-start* and *slug-end* values in a **capi:range-pane**, causing fewer redisplays than if you set each slot directly using its accessor.

13.4.28 Testing for support of display of text and image in menus

The new function **capi:pane-supports-menus-with-images** tests whether menus with an *image-function* display both the images and the text correctly.

13.4.29 Preserving interface state during session saving

If you use session saving in the LispWorks IDE and have your own interface classes, then you might want to preserve their state. To do this, add methods on the new generic function **capi:interface-preserve-state**.

13.5 Other CAPI changes

13.5.1 Cocoa default fonts corrected

CAPI/Cocoa default fonts now match the Apple Human Interface Guidelines (HIG). In LispWorks 5.1 and previous releases, the default fonts used by buttons and various text panes were slightly too small compared to what the Apple HIG specifies.

The font returned by a graphics ports Portable Font Description with **`:stock :system-font`** is now what Apple calls the "system" font. In LispWorks 5.1 and previous releases, it was the "User" font.

13.5.2 editor-pane scroller size

The size of the scroller in a **`capi:editor-pane`** scroll bar now depends on the number of lines which are visible. In previous LispWorks releases, it depended on the number of characters visible, which would cause the scroller to change size noticeably during scrolling.

13.5.3 opengl-pane package change

The OpenGL example now defines **`opengl-pane`** in the OPENGL package rather than the CAPI package. This avoids problems with modifying the CAPI package.

13.5.4 menu items-function called earlier

The *items-function* of a **`capi:menu`** is now called earlier, before the menu is raised (in order to initialize accelerators), and in particular it may be called before the interface is created.

This change may break code which works in LispWorks 5.1 but relies on the interface being displayed at the time *items-function* is called.

13.5.5 Interpretation of repeated initargs and geometric hints

When the initargs passed to make-instance (or the plist passed to **`capi:set-hint-table`**) contain repeated geometric hints like this:

```
(make-instance 'capi:list-pane
       :visible-min-height '(:character 1)
       :visible-min-height '(:character 2))
```

then the first occurrence of each hint keyword is now used to supply the value, that is `(:character 1)` in the example above. This behavior matches the normal interpretation of keyword arguments.

LispWorks 5.1 and earlier versions use the value of the last occurrence of a keyword, that is `(:character 2)` in the example above.

13.5.6 Change preventing premature destruction of a gp:image

Calling **`gp:load-image`** on a **`gp:image`** object now associates the image with the specified port (if the image was not already associated). This means that the image is not destroyed until the port is destroyed or **`gp:free-image`** is called with that port and image.

In LispWorks 5.1 and earlier calling **`gp:load-image`** on a **`gp:image`** can create a situation in which the image is destroyed while it is used by the port. The change prevents this.

13.5.7 Redisplay of image buttons, and change to accessors

The accessors **`capi:button-image`**, **`capi:button-disabled-image`**, **`capi:button-armed-image`**, **`capi:button-selected-image`** and **`capi:button-selected-armed-image`** have been changed. When called after the button has been displayed, these accessors now return the value that was set initially. This change also means that buttons with images can reliably be destroyed and then displayed again.

In LispWorks 5.1 and earlier versions these accessors return the loaded image (that is the result of calling **`gp:load-image`**) and buttons with images cannot reliably be displayed again after being destroyed.

13.5.8 Limiting text in a Cocoa text-input-pane

The **`capi:text-input-pane`** initarg *max-characters* now works on Cocoa. In LispWorks 5.1 *max-characters* has no effect on this platform.

13.5.9 Change in set-rich-text-pane-character-format

capi:set-rich-text-pane-character-format no longer accepts a keyword argument *default*. You can get the same effect by passing **:default** as the value of *attributes-plist*.

13.6 More new features

For details of these, see the documentation in the *LispWorks User Guide and Reference Manual*, unless a manual is referenced explicitly.

13.6.1 Load and evaluate from the command line

The new command line arguments **-load** and **-eval** allow you to specify a file to load, or a form to evaluate, when LispWorks starts. See "The Command line" in the *LispWorks User Guide and Reference Manual* for details of these and all the LispWorks command line arguments.

13.6.2 Unicode character and string functions

13.6.2.1 Unicode case insensitive character comparision

New functions **lw:unicode-char-equal**, **lw:unicode-char-not-equal**, **lw:unicode-char-lessp**, **lw:unicode-char-not-lessp**, **lw:unicode-char-greaterp** and **lw:unicode-char-not-greaterp** compare characters similarly to **cl:char-equal** etc, but using Unicode's simple case folding rules.

13.6.2.2 Unicode case insensitive string comparision

New functions **lw:unicode-string-equal**, **lw:unicode-string-not-equal**, **lw:unicode-string-lessp**, **lw:unicode-string-not-lessp**, **lw:unicode-string-greaterp** and **lw:unicode-string-not-greaterp** compare strings similarly to **cl:string-equal** etc, but using Unicode's simple case folding rules.

13.6.2.3 Unicode character predicates

New predicates `lw:unicode-alphanumericp`, `lw:unicode-alpha-char-p`, `lw:unicode-lower-case-p`, `lw:unicode-upper-case-p` and `lw:unicode-both-case-p` test for properties of a character in Unicode's "general category".

13.6.3 System message log

There is now a system message log recording messages that indicate that something is not as expected, but is not an error. For example, putting a bad Break-Gesture in a GTK resource file.

The function `hcl:set-system-message-log` allows you to manipulate the system message log.

13.6.4 Logging errors

A new API allows your programs to generate a bug form and write it to a log file, optionally printing a message saying that this has been done. See `dbg:log-bug-form` for the details.

Also, `dbg:output-backtrace` is now documented, for use when you want to include Lisp backtraces in log files which you control directly.

Note: Adequate logging of errors is frequently omitted from customer applications, but it is important for the maintenance of any complex application. If your LispWorks application does not log errors via its error handlers, use one of the above APIs to add error logging now!

13.6.5 Debugger command to obtain the current function object

The new debugger command `:func` returns (and sets the value of `*` to) the function object of the current call frame. This is especially useful for call frames of closures and method functions.

13.6.6 Debugger wrapper

The new macro `dbg:with-debugger-wrapper` executes code with a specified function bound as a "debugger wrapper" which takes effect if the code gets an error and tries to invoke the debugger.

Your wrapper function can do whatever is needed, such as locking entry to the debugger in order to prevent multiple processes entering the debugger at once.

13.6.7 Profiling multiple threads

`hcl:do-profiling` is a new convenience function for profiling multiple threads.

13.6.8 Profiler shows each thread separately

When displaying profiler output, the profiler now shows a separate call tree for each thread. The cumulative profile summary is the total for all threads.

13.6.9 Profiler does not count calls by default

On Intel-based platforms, the profiler no longer collects call count information by default.

This has been changed because the counting significantly affects the performance of the application, which skews the results, especially for multi-threaded applications.

13.6.10 Profiling inside foreign calls, or not

You can now specify that programmatic profiling ignores processes that are inside foreign calls. See **`hcl:start-profiling`**.

13.6.11 Long and short forms of paths

Microsoft Windows supports Long and Short forms of paths. You can convert a pathname using one of the new functions **`win32:long-namestring`** or **`win32:short-namestring`**.

13.6.12 Finding a directory for writing temporary files

The function **`hcl:get-temp-directory`** returns a writable directory that can be used for temporary files.

13.6.13 Splitting saved images to allow code signing

The function **save-image** now accepts a keyword argument **:split**, which allows the Lisp heap to be split into a separate file. The main use of split is to allow third-party code signing to be applied to the executable, which is often not possible for an image with the Lisp heap included in a single file.

13.6.14 Saving a Mac OS X application bundle

You can create a Mac OS X application bundle whilst saving a LispWorks image by using the new function **hcl:save-image-with-bundle**.

To simply create an application bundle, use **hcl:create-macos-application-bundle**.

13.6.15 Split sequence utilities

lw:split-sequence and related functions split a sequence into subsequences.

13.6.16 Predicate for weak arrays

The new function **hcl:array-weak-p** tests whether an object is a weak array.

13.6.17 Free action for weak hash tables

There is now an option to add a "free action" to weak hash tables, called after an entry is automatically removed by the Garbage Collector. See **cl:make-hash-table** in the *LispWorks User Guide and Reference Manual*.

13.7 IDE changes

This section describes new features and other changes in the LispWorks Integrated Development Environment (IDE).

See the *LispWorks IDE User Guide* for details of the features mentioned.

13.7.1 ASDF integration

You can now work with ASDF systems in the System Browser tool and Search Files tool in the same way as you can use LispWorks own `lw:defsystem` systems. Various LispWorks editor commands also work on ASDF systems.

To use ASDF in the LispWorks IDE:

1. Load ASDF and some ASDF system definitions in the usual ASDF way.
2. Load the file `(example-file "misc/asdf-integration.lisp")`.

In the tools, an ASDF system named cffi is denoted ASDF:cffi, while an `lw:defsystem` system named foo is denoted LW:foo, so you can distinguish the two types of system definition if necessary.

Note: The interface is intended to be able to accomodate other system definition utilities as well. The comments in `asdf-integration.lisp` explain how it works in detail.

13.7.2 Break gesture available in the Cocoa IDE

On Cocoa the gesture `Command+Ctrl+,` (comma) is the break gesture.

This is useful if the "spinning beach ball" cursor appears for some time and LispWorks appears to be hanging.

13.7.3 Enhanced Break gesture handling

When the break gesture is used, LispWorks attempts to break the current operation in a useful way, allowing you to regain control even if the GUI is busy. It attempts to find a busy process to break. If there is no obvious candidate and the LispWorks IDE is running, then it displays the Process Browser tool.

13.7.4 Mac OS X toolbars

On Cocoa, the LispWorks tools now have standard Mac OS X toolbars.

You can hide or show these toolbars using the button at the top right corner of the title bar, and modify them using the standard commands on the context menu.

13.7.5 Customizable toolbars

On all platforms you can now select which toolbar buttons are displayed for each tool in the LispWorks IDE. To do this, select **Customize** on the context menu of the toolbar.

13.7.6 Preferences consolidated

The IDE preference options are now all shown in a single Preferences window which is easier to use than the separate global and per-tool preference dialogs seen in LispWorks 5.1.

Raise the Preferences window by **Tools > Preferences...**. On Cocoa, press **Command+,** (comma) or select **LispWorks > Preferences...**.

The Preference options you choose affect all tools including those currently on screen, as well as those created later and after restarting LispWorks.

13.7.7 Editor font preference changed

You now select the Editor font preference option by **Preferences... > Environment > Styles > Editor Font**.

The Editor Font preference takes effect only if you select **Override the system default font**. Then an area displaying sample text in the current preference font become active, and you can choose a new font by double-clicking on the sample text.

The Editor Font preference setting now affects all Editor tools and other tools based on `capi:editor-pane` such as the Listener tool.

There are no longer options for applying the setting only to current Editor windows or the current window only, as in LispWorks 5.1.

13.7.8 Parenthesis coloring

The Editor now displays Lisp code with matching pairs of parentheses colored acording to their depth within the current top level form.

You can switch parenthesis coloring on and off by **Preferences... > Environment > Styles > Color parenthesis**, and you can configure the colors used with `capi:set-editor-parenthesis-colors`.

13.7.9 Snapshot debugging on initialization

The new variable `*debug-initialization-errors-in-snap-shot*` controls whether, in an image which is configured to start the LispWorks IDE automatically, an error during initialization is handled and displayed in a snapshot debugger after the IDE starts.

13.7.10 The default directory for opening files

Operations such as **File > Open** now, by default, use the directory of the file most recently edited in the IDE as the default directory in the file dialog.

This behavior is controlled by the new preference option **Preferences... > Environment > General > Use recent directory for opening files**.

If this option is deselected, the current working directory will be used, which is the behavior seen in LispWorks 5.1 and previous versions.

Note that this does not apply to the Editor tool itself, for which the file dialog always uses the directory of the currently visible file as the default directory.

13.7.11 Prompt on exit for modified buffers not associated with a file

By default the IDE now prompts for confirmation on exit when there is a modified buffer which is not associated with a file, such as one created by **File > New** or the editor command `New Buffer`.

This behavior is controlled by the option **Preferences... > Environment > General > Confirm Before Exiting > When modified buffers**.

In LispWorks 5.1 this option would cause a prompt only for buffers associated with files.

13.7.12 Scroller size in Editor and other tools

By default, the size of the scroller in an Editor tool's scroll bar now depends on the number of lines which are visible. This change also applies to other tools based on `capi:editor-pane`, such as the Output Browser and Shell.

In LispWorks 5.1 and earlier versions the scroller size depends on the number of characters visible, which can cause it to change size greatly during scroll-

ing. The method of calculating the scroller size can be reverted to that used in LispWorks 5.1 by the new editor command **Toggle Count Newlines**.

13.7.13 Editor status bar shows line numbers

The Editor tool's status bar now shows the range of line numbers currently displayed, along with the total number of lines in the displayed editor buffer. This is configurable, as described in "Position indicator shows line numbers" on page 141.

13.7.14 New and changed Edit menu operations

The **Edit** menu operations **Copy**, **Cut**, **Paste**, **Select All**, **Deselect All**, **Find**, **Find Next** and **Replace** now work on the active pane rather than the "primary object" in the tool. The **Edit > Object** submenu allows you to cut, copy and paste the primary object. For more information see "Performing editing functions" in the *LispWorks IDE User Guide*.

13.7.15 Standard Edit gestures on Cocoa

On Cocoa, LispWorks now adds a minimal Edit menu to all CAPI interfaces when running in the IDE, which makes Edit gestures **Command+X**, **Command+C** and **Command+V** work in every interface displayed in the LispWorks IDE. To remove the IDE's automatic menu, pass **:auto-menus nil** initargs when making the interface.

Note: to implement these gestures in your CAPI/Cocoa runtime application, you must include an **Edit** menu explicitly in your interface definition.

13.7.16 More control over Process Browser automatic updates

You can now specify a delay after each automatic update of the Process Browser tool. This has been added because because immediate automatic updates are often too frequent to be useful, and can be expensive.

To set the delay period, first ensure automatic updates are enabled via the **Update Frequency** preference option, and then set the new Process Browser preference option **Automatic Update Delay**.

13.7.17 Search Files with known definitions

The Search Files tool now allows searching amongst all files known to contain definitions, as specified by the value of `dspec:*active-finders*`.

Select **Known Definitions** from the dropdown list in the toolbar to perform this kind of search.

13.7.18 Search Files option for relative or full paths

You can now control how the name of each matching file is shown in the result of the Search Files tool, via the option **Preferences... > Search Files > Display > Files shown**.

13.7.19 Search Files highlights first match for file

After you double-click on a filename in the Search Files tool results area, the cursor moves to the first match in that file when it is displayed in the Editor tool.

13.7.20 Symbol Browser sortable

The main list displayed by the Symbol Browser tool is now sortable. Click on the header of either the **Home Package** or **Name** column.

13.7.21 Completion in Tracer

The Tracer tool now does in-place completion on the symbol to trace. Press `Up` or `Down` in the **Trace:** pane to complete the text already entered.

13.7.22 Inspecting the function in a stack frame

The **Debug** menu in the Debugger tool and the Listener tool now allows you to display an Inspector tool showing the selected frame's function, by **Debug > Frame > Inspect Function**.

13.7.23 Bad interaction with pretty printer eliminated

Display of forms in lists such as method dspecs in the Generic Function Browser no longer interact with the pretty printer. The complete form is now always displayed on one line.

In LispWorks 5.1 and previous releases, setting the global value of **`*print-pretty*`** to **`t`** could cause the display to be truncated.

13.7.24 Removal of breakpoints

Edit Breakpoints > Remove now closes the dialog immediately when all breakpoints have been removed. Thanks to Nick Levine for suggesting this improvement.

13.7.25 Output Browser accelerator

There is now an accelerator key which raises the Output Browser tool.

See "Tool accelerator keys" in the *LispWorks IDE User Guide* for the full list of tool accelerators.

13.7.26 Listen operations move Listener point

Listen operations which put a value in **`*`**, such as **Slots > Listen** in the Inspector tool, now move the cursor to the Listener prompt automatically if it is not already there. This is useful because that is usually where you need to type next.

13.8 Editor changes

This section describes new features and other changes in the LispWorks editor, which is used in the Editor tool of the LispWorks IDE.

See the *LispWorks Editor User Guide* for details of these changes.

13.8.1 Change to Tab key in Lisp mode

The **Tab** key is now overridden in Lisp mode to invoke the new command **Indent Selection or Complete Symbol** (described in "New commands" on page 139).

13.8.2 New commands

The default key bindings mentioned in this section apply to Emacs emulation.

Indent Selection or Complete Symbol does Lisp indentation if there is a visible region. Otherwise, it attempts to indent the current line. If the current line is already indented correctly then it attempts to complete the symbol before the current point, like **Complete Symbol**. In Lisp mode **Indent Selection or Complete Symbol** is bound to **Tab**.

Indent or Complete Symbol attempts to indent the current line. If the current line is already indented correctly then it attempts to complete the symbol before the current point, like **Complete Symbol**.

Function Arglist Displayer shows or hides information about the operator in the current form. The command controls display of a special window (displayer) on top of the editor. The displayer shows the operator and its arguments, and tries to highlight the current argument (that is, the argument at the cursor position). If it does not recognize the operator of the current form, it tries the surrounding form, and if that fails it tries a third level of surrounding form. **Function Arglist Displayer** is bound to **Ctrl+`**. Additionally, while the displayer is visible:

Ctrl+/	controls whether the documentation string of the operator is also shown
Ctrl++	moves the displayer up
Ctrl+-	moves the displayer down

Invoke Tool activates or creates a tool in the LispWorks IDE, prompting for a shortcut character corresponding to the desired tool. **Invoke Tool** is bound to **Ctrl+#**. Tip: if you cannot remember the shortcut character, enter **Ctrl+# ?** to raise the **Tools** menu, which displays the shortcuts.

Make Directory prompts for a directory name and makes the directory in the filesystem.

Insert Double Quotes For Selection inserts a pair of double quotes around the selected text "like this". **Insert Double Quotes For Selection** is bound to **Meta+"**.

Insert Multi Line Comment For Selection inserts multi-line comment markers around the selected text #|like this|#. **Insert Multi Line Comment For Selection** is bound to **Meta+#**.

Insert Parentheses For Selection inserts a pair of parentheses around the selected text (like this). **Insert Parentheses For Selection** is bound to **Meta+(**.

Toggle Count Newlines controls the size of the scroller in editor-based tools, and how the Editor tool's status bar represents the extent of the displayed part of the buffer. When counting newlines, the status bar shows the range of line numbers currently displayed, along with the total line count. When counting characters, the status bar shows the range currently displayed as precentages of the total number of characters in the buffer.

Mark Word marks the word following the current point. A prefix argument, if supplied, specifies the number of words marked. **Mark Word** is bound to **Meta+@**.

Diff Ignoring Whitespace compares the current buffer with another file, ignoring whitespace. A prefix argument, if supplied, compares any two files.

Next Breakpoint and **Previous Breakpoint** move the point to the next or previous breakpoint in the current buffer. A prefix argument *p*, if supplied, specifies that *p*-1 breakpoints are skipped.

Describe Method Call displays a Generic Function Browser tool, with a specific method combination shown. When invoked with a prefix argument while the cursor is in a **defmethod** form, **Describe Method Call** uses the generic function and specializers of the method to choose the method combination. Otherwise, it prompts for the generic function name and the list of specializers.

Build Application invokes the Application Builder tool and does a build. By default, it uses the current buffer as the build script. If a prefix argument is supplied it prompts for a file to use as the build script.

Find Unwritable Character finds the next occurrence of a character in the current buffer that cannot be written using the buffer external format.

List Unwritable Characters lists the characters in the current buffer that cannot be written with the buffer external format.

13.8.3 New echo area commands

Ctrl+C Ctrl+C invokes the new command **Insert Selected Text** which inserts the editor window's selected text in the echo area.

You can now add the editor window's selected text to an incremental search string with the key sequence **Ctrl+S Ctrl+C**.

13.8.4 Position indicator shows line numbers

By default the editor's position indicator (displayed in the status bar of the Editor tool) now shows the range of line numbers currently displayed, along with the total number of lines in the buffer, like this:

```
<StartLine>-<EndLine>[totalLines]
```

The editor can be changed to count characters rather than lines by the command **Toggle Count Newlines**. It then displays percentages rather than line numbers.

13.8.5 Popping marks rotates the mark ring like GNU Emacs

The Editor commands **Set Mark** with a prefix argument, **Pop and Goto Mark** and **Pop Mark** and the function **editor:current-mark** with the *pop-p* argument now rotate the mark ring instead of discarding the current mark. This makes them compatible with GNU Emacs and allows you to cycle around all of the marks by repeated invocation. The word "pop" in the names is retained for compatibility, even though the operation is now a rotation.

13.8.6 File completion shows trailing slash for directories

Completion lists displayed by commands such as **`Find File`** now display the trailing `/` (slash) on directory names, which makes it possible to distinguish them from regular files.

13.8.7 Escaped symbols recognized correctly

Escaped symbols such as `|BAZ-foobar|` are now recognized correctly by commands such as **`Find Source`** which operate on the symbol at the current point.

13.8.8 prompt-for-file :direction and :create-directories

`editor:prompt-for-file` now takes a **`:direction`** argument. You can specify *direction* **`:input`** (when expecting to read the file) or *direction* **`:output`** (when expecting to write the file).

`editor:prompt-for-file` also has a new **`:create-directories`** argument. If *create-directories* is true, then the user is prompted to create any missing directories in the path she enters.

13.8.9 Definers for editor variables

The new macro **`editor:define-editor-variable`** defines (or sets the value of) a global editor variable.

The new macro **`editor:define-editor-mode-variable`** defines (or sets the value of) an editor variable in the specified editor mode.

The main advantages of these macros are that they:

- provide a way of defining editor variables which is more readable than **`(setf editor:variable-value)`**
- are recognized by the source location system including commands such as **`Find Source`**.

13.8.10 Buffer variable value

The new function **`editor:buffer-value`** accesses the value of an editor variable in the specified buffer.

13.8.11 New exports

The functions **editor:get-symbol-from-point** and **editor:point>=** are now exported.

13.9 Foreign Language interface changes

See the *LispWorks Foreign Language Interface User Guide and Reference Manual* for details of these changes.

13.9.1 Using arbitrary Lisp integers in the FLI

A new mechanism allows you to convert any Lisp integer into a foreign array of bytes and to convert that array back to an equivalent Lisp integer. This would allow the integer to be stored in and retrieved from a database, for example.

See the section "Lisp Integers", the macro **fli:with-integer-bytes** and the functions **fli:convert-integer-to-dynamic-foreign-object** and **fli:make-integer-from-bytes** in the *LispWorks Foreign Language Interface User Guide and Reference Manual*

13.9.2 Enum type values and symbols

New functions **fli:enum-values**, **fli:enum-symbols** and **fli:enum-symbol-values-pairs** return the values and symbols for a FLI enumerator type specifier which was defined by **fli:define-c-enum**.

13.9.3 :ignore in define-foreign-function lambda list

fli:define-foreign-function has a new option **:ignore** in the syntax for an argument which is useful for **:reference-return** arguments. It specifies that **nil** is always passed through to the foreign code and the argument is omitted from the lambda list of the Lisp function.

13.9.4 define-foreign-variable with aggregate types

The implementation of **fli:define-foreign-variable** has changed to require the **:accessor** argument if the type is an aggregate type. This is to

avoid accidental copying of aggregates. To retain the existing copying semantics, use `:accessor :value`.

13.9.5 Pointer and array type information

The functions `fli:pointer-element-size` and `fli:pointer-element-type` have been extended to accept a FLI pointer type as argument, and return the size (or type) of the elements of that type.

The functions `fli:foreign-array-dimensions` (and `fli:foreign-array-element-type`) have been extended to accept the name of a FLI array type as argument and in this case return the dimensions (or type of the elements) of the array type.

13.9.6 New foreign types

This section describes the new FLI types in LispWorks 6.0.

13.9.6.1 FLI types for ISO C size_t and ptrdiff_t

New foreign types `:size-t` and `:ptrdiff-t` convert between a Lisp integer and an ISO C size_t or ISO C ptrdiff_t.

13.9.6.2 ISO C99 types

New foreign types are defined for integers of particular sizes. These are equivalent to the types defined by ISO C99. For example, Lisp `:uint8` is C99 uint8_t. The new foreign types are:

`:int8`	8-bit signed integer
`:uint8`	8-bit unsigned integer
`:int16`	16-bit signed integer
`:uint16`	16-bit unsigned integer
`:int32`	32-bit signed integer
`:uint32`	32-bit unsigned integer
`:int64`	64-bit signed integer

`:uint64`	64-bit unsigned integer
`:intmax`	largest type of signed integer available
`:uintmax`	largest type of unsigned integer available
`:intptr`	signed integer the same size as a pointer
`:uintptr`	unsigned integer the same size as a pointer

13.9.6.3 FLI type for ssize_t

The FLI `:ssize-t` type converts between a Lisp integer and a platform-specific ssize_t type, which is a signed integer representing the size of an object in bytes.

13.9.6.4 FLI type for ISO C time_t

The FLI `:time-t` type converts between a Lisp integer and an ISO C time_t type, which is an integer type used for storing system time values.

13.9.7 define-foreign-function result-type :void

The function defined by `fli:define-foreign-function` (or `fli:define-foreign-funcallable`) now returns no value when *result-type* is `:void`. In previous releases, the value `nil` was returned. In practice, this change only affects cases where additional values are returned due to arguments with `:reference` or `:reference-return` types.

13.9.8 define-foreign-converter documented

The macro `fli:define-foreign-converter` is now fully documented. It defines a new FLI type specifier that converts to and / or from another type specifier.

13.9.9 Deprecated macro

The macro `fli:with-dynamic-foreign-objects-with-cleanups` is deprecated.

13.10 COM/Automation changes

This section applies only to Windows platforms. See the *LispWorks COM/Automation User Guide and Reference Manual* for details.

13.10.1 Building a COM server

You can now build a DLL with the standard symbols that a COM server needs, by passing `:com` as the *dll-exports* in `deliver` or `save-image`.

13.10.2 com:get-object

The new function `com:get-object` returns an interface pointer for a named object.

13.10.3 Automation server example

A new example shows how you can build a CAPI application controlled by Automation, and an Automation server to control it.

See the files in `examples/com/automation/capi-application/`.

13.11 Objective-C changes

13.11.1 Creating a standalone Objective-C executable

A new example demonstrates how to deliver the AreaCalculator example as a standalone executable. See `examples/objc/area-calculator/deliver.lisp`.

13.12 Common SQL changes

13.12.1 Refreshing select recomputes deferred join slots

Passing *refresh* `t` to an object-oriented `sql:select` clause now causes any join slots defined using *retrieval* `:deferred` to be recomputed the next time time they are accessed.

13.12.2 Oracle on 64-bit Mac OS X

64-bit LispWorks now supports Oracle 10.2.0.4 on Intel Mac OS X.

At the time of writing there no 64-bit PowerPC Oracle instant client library is available, therefore 64-bit LispWorks cannot support Oracle on PowerPC Macintoshes.

13.12.3 PostgreSQL TIMESTAMP and sql:universal-time

PostgreSQL TIMESTAMP columns are now converted to Lisp universal values when **`sql:universal-time`** is specified as the type in **`sql:def-view-class`**.

13.13 Application delivery changes

See the *LispWorks Delivery User Guide* for more details of the changes mentioned in this section.

13.13.1 Splitting delivered images to allow code signing

The function **`deliver`** now accepts a keyword argument **`:split`**, which allows the Lisp heap to be split into a separate file. The main use of split is to allow third-party code signing to be applied to the executable, which is often not possible for an image with the Lisp heap included in a single file.

13.13.2 Application control over multiprocessing startup

Delivery now allows the application to control startup of multiprocessing. In previous LispWorks releases, only automatic startup of multiprocessing was supported.

See the new value **`:manual`** for the **`:multiprocessing`** keyword argument to **`deliver`**.

13.13.3 GTK+ is the default GUI on some platforms

On Linux and FreeBSD (and the X11 GUI option on Mac OS X), the distributed image now uses GTK+ rather than Motif. This also applies to your delivered

application unless you modify your build script, as described in “Building Motif applications” on page 148.

13.13.4 Building Motif applications

To build a Motif application on Linux, x86/x64 Solaris and FreeBSD (and when using the X11 GUI option on Mac OS X), add

```
(require "capi-motif")
```

to the build script.

For all platforms, you should ensure that Motif is installed on the target machine. Additionally, if your application uses image formats other than BMP, you should ensure that Imlib is installed on the target machine.

13.13.5 Control of make-instance initarg checking simplified

The **:make-instance-keyword-check** keyword in **deliver** is now processed regardless of the setting of the **:keep-clos** keyword.

In addition, its default behavior is now to retain the current setting of **make-instance** checks when **:keep-debug-mode** is used, rather than forcing the checks to be switched on.

13.14 CLOS/MOP changes

cl:update-instance-for-redefined-class and **cl:update-instance-for-different-class** lock the redefined instance against access whilst updating, so your methods should take care to avoid deadlocks. See the reference entries in the *LispWorks User Guide and Reference Manual*.

The function **clos:compute-default-initargs** is now exported for compatibility with AMOP.

13.15 CLIM changes

13.15.1 Default I/O streams corrected.

The following CLIM macros have been changed to match the documentation if the stream variable is not specified:

accept-values-command-button, **accepting-values**, **with-input-editing**, **with-input-editor-typeout** and **completing-from-suggestions** now default to ***query-io***. In LispWorks 5.1 and previous versions, the default was ***standard-input***.

dragging-output and **tracking-pointer** output now defaults to ***standard-output***. In LispWorks 5.1 and previous versions, the default was ***standard-input***.

13.16 Other changes

13.16.1 Switching off Windows themes before windows are made

The call **(win32:set-application-themed nil)** in LispWorks for Windows now works when the call is made before any windows have been opened.

LispWorks 5.x programs which make an early call **capi:convert-to-screen** or **clim:find-port** to initialize windowing solely for the purpose of enabling **win32:set-application-themed** no longer need to do so in LispWorks 6.0.

13.16.2 Non-string argument in find-symbol and intern

Calling **cl:find-symbol** or **cl:intern** with a first argument which is not a string now signals error, as required by ANSI Common Lisp. LispWorks 5.1 and previous versions erroneously allowed a symbol argument.

13.16.3 Compiler messages are written to *error-output*

Warning messages from **cl:compile** and **cl:compile-file** now go to the stream that is the value of **cl:*error-output***. This is consistent with the passed X3J13 cleanup issue COMPILER-WARNING-STREAM and also

matches some other Common Lisp implementations. In previous LispWorks releases, such messages were written to **cl:*standard-output***.

13.16.4 Re-reading from terminal stream on Cocoa fixed

On Cocoa, reading from **cl:*terminal-io***, closing Terminal.app and then reading again opens a new Terminal.app window. In LispWorks 5.1 and previous versions, an end of file error is signalled.

13.16.5 Explaining failed funcall optimizations

A new compiler explanation, available via **(declare (:explain ...))**, tells you when a funcall optimization fails because of lack of type information.

13.16.6 Change in make-random-state

(cl:make-random-state t) now uses **/dev/urandom**, if available, and only if this is not available does it resort to **/dev/random**.

The reason for this change is that **/dev/random** pauses when it runs out of entropy, so code with multiple calls can be slow.

13.16.7 Escaping of printed symbols depends on macro chars

When printing symbols any characters in the package name or symbol name that have a reader macro in the current readtable are now escaped, as required by the ANSI Common Lisp standard.

13.16.8 dotimes count type-checked

cl:dotimes now always checks the type of its *count* argument. In LispWorks 5.1 and earlier versions, there was a type check only in code compiled at **debug 3**.

13.16.9 Changes in cl:*features*

For a full description including information about the features used to distinguish new LispWorks implementations and platforms, see the entry for **cl:*features*** in the *LispWorks User Guide and Reference Manual*.

13.16.10 Gesture Spec changes

See **sys:make-gesture-spec** for details of the changes listed in this section.

13.16.10.1 Gesture Spec keyword changes

The following changes apply to the keywords which can be passed as the *data* in **sys:make-gesture-spec**:

- **:prev** is now **:prev-item**
- **:next-item** is new.
- On X11, "Sys_Req" now generates **:sys-req** (like it already does on Cocoa in LispWorks 5.1).

Also, the full set of keywords which **sys:make-gesture-spec** accepts are now documented.

13.16.10.2 Combining both-case-p characters with Shift

In order to ensure a consistent representation of **both-case-p** characters, Gesture Specs representing these were designed such that they may not have a single modifier **Shift**. This restriction can now be overridden.

13.16.11 64-bit OpenSSL on Mac OS X

A 64-bit OpenSSL library is now installed by default in all versions of Mac OS X that LispWorks supports, therefore you should use this rather than any other version of 64-bit OpenSSL.

If you think you might be using a non-standard OpenSSL library, check for calls to **comm:set-ssl-library-path**.

13.16.12 Lambda lists corrected

The lambda lists of **appendf** and other users of **define-modify-macro** have now been corrected.

Lambda lists previously recorded as **&rest args** for special forms like **cl:when** and **cl:unless** have also been corrected.

Lambda lists are reported by commands like **Function Lambda List** and can be seen interactively by using **examples/editor/show-arglist.lisp**.

13.16.13 Loading old data files

Binary files created with **hcl:dump-forms-to-file** or **hcl:with-output-to-fasl-file** in LispWorks 5.1, LispWorks 5.0, LispWorks 4.4 or LispWorks 4.3 can be loaded into LispWorks 6.0 using **sys:load-data-file**.

13.16.14 Deprecated APIs

hcl:set-promotion-count is deprecated, because experience has shown that it is not useful.

13.16.15 Harlequin

Long term users may be interested to know that there is a Harlequin in LispWorks 6.0, which was not present in previous versions. Can you spot it?

13.17 Documentation changes

The HTML documention now contains breadcrumbs at the top of each content page. These aid navigation.

The *LispWorks User Guide* and *LispWorks Reference Manual* are now consolidated into the combined *LispWorks User Guide and Reference Manual*, with cross-references.

The *LispWorks Editor User Guide* now includes documentation for some editor variables that you may want to change.

The printable manuals are PDF Version 1.5. You may need to upgrade to a compatible PDF reader.

13.18 Known Problems

13.18.1 Runtime library requirement on Windows

LispWorks for Windows requires the Microsoft Visual Studio runtime library **msvcr80.dll**. The LispWorks installer installs this DLL if it is not present.

Applications you build with LispWorks for Windows also require this DLL, so you must ensure it is available on target machines.

13.18.2 Problems with CAPI on GTK+

The **capi:interface-override-cursor** is ignored by **capi:text-input-pane** which always displays its usual I-beam cursor. This is due to a limitation in the way that text-input-pane is implemented by GTK.

The normal navigation gesture (**Tab**) is treated as an editor command in **capi:editor-pane** and IDE tools based on this. Instead, use **Ctrl+Tab** to navigate from an editor pane in GTK+.

In GTK+ versions older than 2.12, the value of **capi:option-pane** *enabled-positions* has no effect on the visible representation of the items. In later versions of GTK+, the disabled items are grayed out.

13.18.3 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 **cl:*terminal-io*** (which is not normally visible). Possibly when the IDE is running, output should be connected to the Background Output buffer.

13.18.4 Problems with the LispWorks IDE on Cocoa

Multithreading in the CAPI is different from other platforms. In particular, all windows run in a single thread, whereas on other platforms there is a thread per window.

The debugger currently doesn't work for errors in Cocoa Event Loop or Editor Command Loop threads. However, there is a **Get Backtrace** button so you can obtain a backtrace and also a **Debug Snapshot** button which aborts from the error but displays a debugger with a copy (snapshot) of the stack where the error occurred.

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

Setting `*enter-debugger-directly*` to `t` can allow the undebuggable processes to enter the debugger, resulting in the UI freezing.

Inspecting a long list (for example, 1000 items) via the Listener's `Inspect Star` editor command prompts you about truncation in a random window. If you cancel, the inspect is still displayed.

The editor's Help about help (`Control+h Control+h`) dialog is messy because it assumed that a fixed width font is being used.

The **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.

13.18.5 Problems with CAPI and Graphics Ports on Cocoa

The `capi:interface-override-cursor` is ignored.

Some graphics state parameters are ignored, in particular operation, stipple, pattern, fill-style and mask (other than a rectangle).

LispWorks ignores the System Preferences setting for the smallest font size to smooth.

There is no support for state images or checkboxes in `capi:tree-view`.

`capi:with-page` does not work, because Cocoa tries to control page printing.

The `:help-callback` initarg is only implemented for the `:tooltip` value of the type argument.

The **:visible-border** initarg only works for scrolling panes.

Caret movement and selection setting in **capi:text-input-pane** is implemented, but note that it works only for the focussed pane.

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

There is no meta key in the input-model of **capi:output-pane**. Note that, in the editor when using Emacs emulation, the **Escape** key can be used as a prefix.

There has been no testing with 256 color displays.

There is no visual feedback for dead-key processing, for example **Option+n** is the tidle dead-key.

The graph pane's plan mode rectangle doesn't redraw when moved or resized.

Some pinboard code uses **:operation boole-xor** which is not implemented.

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. Unfortunately Mac OS X fonts do not generally guarantee that, so for example Monaco 10, 15, 20 pt can be used etc but not Monaco 12 pt. The nearest good size is used instead, so as another example you might select Menlo 11-pt, but editor text would be displayed in Menlo 12-pt.

The default menu bar is visible when the current window has no menu bar.

capi:tree-view is slow for a large number (thousands) of items.

The editor displays decomposed characters as separate glyphs.

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

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

13.19 Binary Incompatibilty

If you have binaries (fasl files) which were compiled using LispWorks 5.1 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 6.0

Index

D

E

F

M

N

O

P

R

S

T

U

V

W