LispWorks®

Release Notes and Installation Guide

Version 6.1

Copyright and Trademarks

LispWorks Release Notes and Installation Guide

Version 6.1

December 2011

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 3.

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 6.1 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 6.1 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

On most platforms you can choose either the 32-bit or 64-bit implementation of LispWorks Enterprise Edition. These are separately licensed.

1.2 LispWorks for UNIX

On SPARC Solaris and HP-UX the Edition model described above does not apply to 32-bit LispWorks. LispWorks 6.1 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.

LispWorks Enterprise (64-bit) for Solaris is separately available and follows the "Enterprise Edition" licensing model.

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 6.1 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 6.1. 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 & 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 6.1 (32-bit) for Macintosh and LispWorks 6.1 (64-bit) 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 the 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 6.1/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 6.1/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 6.1 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, 10.6.x or 10.7.x GTK+ 2 (version 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, 10.6.x or 10.7.x. GTK+ 2 (version 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 6.1. There may be a downloadable patch bundle which upgrades LispWorks to version 6.1.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 6.1.

2.4.2 Information for Beta testers

Users of LispWorks 6.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.1.

See “Uninstalling LispWorks for Macintosh” on page 14 for instructions.

2.4.3 Information for users of previous versions

You can install LispWorks 6.1 in the same location as LispWorks 6.0 or previous versions. If you always choose the default install location, a new **`LispWorks 6.1`** folder will be created alongside the other versions.

Similarly LispWorks Personal Edition 6.1 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-6.1.dmg`** or **`LispWorks64bit-6.1.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 6.1`, `LispWorks 6.1 (64-bit)` or `LispWorks Personal 6.1` 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 (version 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 6.1 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-6-1-0-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-6-1-0-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 distinct 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 6.1"
  xterm$ ./lispworks-6-1-0-macos-universal-gtk
  LispWorks(R): The Common Lisp Programming Environment
  Copyright (C) 1987-2011 LispWorks Ltd.  All rights reserved.
  Version 6.1.0
  Saved by LispWorks as lispworks-6-1-0-x86-darwin-gtk, at 13 Jan
  2011 1:55
  User dubya on octane
  ; Loading text file /Applications/LispWorks 6.1/Library/lib/6-1-
  0-0/config/siteinit.lisp
  ;  Loading text file /Applications/LispWorks 6.1/Library/lib/6-1-
  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 6.1"
  xterm% ./lispworks-6-1-0-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-6-1-0-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-6-1-0-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 Uninstalling LispWorks for Macintosh

To uninstall LispWorks you can either:

- Launch the LispWorks installer as described in “Launch the LispWorks installer” on page 8 and select the **Uninstall** option (and then remove any patches), or

- Simply drag the `LispWorks 6.1` folder to the trash.

2.7 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 6.1 (32-bit) for Windows and LispWorks 6.1 (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 6.1 > 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 6.1 > 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 6.1. There may be a downloadable patch bundle which upgrades LispWorks to version 6.1.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 6.1.

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 6.1 in the same location as LispWorks 6.0, LispWorks 5.x or LispWorks 4.4.5. This is the default installation location.

You can also install LispWorks 6.1 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 6.1 Beta should completely uninstall it before installing LispWorks 6.1. Remember to remove any patches added since the Beta release.

See “Uninstalling LispWorks for Windows” on page 20 for instructions.

3.2.5 To install LispWorks

To install LispWorks (32-bit) for Windows run **`x86-win32\LispWorks61-32bit.exe`**.

To install LispWorks (64-bit) for Windows run **`x64-windows\LispWorks61-64bit.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 accessible 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.lispworks.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 6.1 > 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 Uninstalling LispWorks for Windows

To uninstall LispWorks:

1. Run **Add or Remove Programs** (Windows XP) or **Programs and Features** (Windows Vista / Windows 7).
2. Select **LispWorks 6.1** and click **Remove**.

This will uninstall LispWorks along with any installed updates. It will not remove any private patches.

3.4 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 6.1 (32-bit) for Linux and LispWorks 6.1 (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 2.3.3 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 (version 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.	Firefox or Opera web browser for viewing on-line documentation

Table 4.1 System requirements on Linux

4.1.1 GUI libraries

LispWorks 6.1 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+ 2 (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 6.1 Motif GUI and LispWorks 6.0 or 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.lispworks.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 6.1 for Linux is supplied as a download. A back-up CD-ROM is available. 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 back-up 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 6.1. There may be a downloadable patch bundle which upgrades LispWorks to version 6.1.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 6.1.

4.4.2 Information for Beta testers

Users of LispWorks 6.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.1.

See "Uninstalling LispWorks for Linux" on page 33 for instructions.

4.4.3 Installation from the binary RPM file

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

If you already have LispWorks 6.1 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-6-1-0-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-6-1-0-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 105 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-6.1-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-6.1-*.i386.rpm` wherever `lispworks-6.1-*.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-6.1-`***n***`.i386.rpm`.
2. Install or upgrade LispWorks in the standard RPM way, for example:

   ```
   rpm --install lispworks-6.1-n.i386.rpm
   ```

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

   ```
   rpm --install --prefix <directory> lispworks-6.1-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-6.1-`*`n`*`.i386.rpm`	CLIM 2.0

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

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-6.1-`*`n`*`.i386.rpm`	CLIM 2.0

Table 4.3 File distributions for layered products in the Enterprise Edition

File Distribution	Layered Product
`lispworks-kw-6.1-n.i386.rpm`	KnowledgeWorks
`lispworks-corba-6.1-n.i386.rpm`	LispWorks ORB
`lispworks-sql-6.1-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/6-1-0-0/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-6.1-n.i386.rpm
```

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

```
rpm --install --prefix <directory> --replacepkgs
lispworks-6.1-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

`lw61-x86-linux.tar.gz`	32-bit LispWorks image, modules and examples
`lwdoc61-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-6-1-0-x86-linux`	Personal Edition
`lispworks-6-1-0-x86-linux`	32-bit Professional or Enterprise Edition
`lispworks-6-1-0-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 101 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-6-1-0-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 **lisp-keys@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 & FreeBSD”.

4.8 Printable LispWorks documentation

In a default installation, the `lib/6-1-0-0/manual/offline` directory contains PDF format versions of the manuals.

These files are also available 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 this command, as root:

```
rpm --erase lispworks-6.1
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named `lispworks-patches6.1`. The same applies to additional RPM packages such as `lispworks-sql`.

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 6.1 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), Solaris 11, or Open-Solaris (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 (version 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.	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 6.1 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+ 2 (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.lispworks.com/documentation`** in any case, and the same manuals are also available there in PostScript format.

5.2 Software on the CD-ROM

LispWorks 6.1 for x86/x64 Solaris is supplied as a standard package file to download. A back-up CD-ROM is available. 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 **`LispWorksPersonal61-32bit`**.

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 6.1. There may be a downloadable patch bundle which upgrades LispWorks to version 6.1.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 6.1.

5.3.2 Information for Beta testers

Users of LispWorks 6.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.1.

See “Uninstalling LispWorks for x86/x64 Solaris” on page 41 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-6-1-0-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-6-1-0-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-6-1-0-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 `LispWorks61-32bit` and can be found in the `x86-solaris` directory of the LispWorks 6.1 CD-ROM. The package name is `LispWorks61-32bit`.

The LispWorks (64-bit) Enterprise software package file is called `LispWorks61-64bit` and can be found in the `amd64-solaris` directory of the LispWorks 6.1 CD-ROM. The package name is `LispWorks61-64bit`.

The Personal Edition software package file is called `LispWorksPersonal61-32bit`. The package name is `LispWorksPersonal61-32bit`.

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 LispWorks61-32bit all
```

for 32-bit LispWorks, or

```
pkgadd -d LispWorks61-64bit 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

Run LispWorks (all variants) from the directory **`/opt/LispWorks/bin`**.

The LispWorks executable is named as shown here:.

`lispworks-personal-6-1-0-x86-solaris`	Personal Edition
`lispworks-6-1-0-x86-solaris`	Professional or Enterprise Edition
`lispworks-6-1-0-amd64-solaris`	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.

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 101 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-6-1-0-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 & FreeBSD”.

5.7 Printable LispWorks documentation

In a default installation, the **`lib/6-1-0-0/manual/offline`** directory contains PDF format versions of the manuals.

These files are also available at **`www.lispworks.com/documentation/`**.

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

5.8 Uninstalling LispWorks for x86/x64 Solaris

To uninstall LispWorks, perform the following steps as root:

1. If patches for LispWorks 6.1 have been installed then you will need to uninstall the patch package, by

   ```
   pkgrm -n LispWorksPatches61-32bit
   ```

   or

   ```
   pkgrm -n LispWorksPatches61-64bit
   ```

2. Then uninstall the main software package containing LispWorks 6.1 by executing:

   ```
   pkgrm -n LispWorks61-32bit
   ```

   or

   ```
   pkgrm -n LispWorks61-64bit
   ```

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 6.1 (32-bit) for FreeBSD and LispWorks 6.1 (64-bit) 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 (32-bit) plus documentation	FreeBSD 7.x, or later with compat7x (if you want to run 32-bit LispWorks on FreeBSD 6.x, then please contact Lisp Support)

Table 6.1 System requirements on FreeBSD

Hardware Requirements	Software Requirements
180MB of disk space for Enterprise Edition (64-bit) plus documentation	GTK+ 2 (version 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.	Firefox or Opera web browser for viewing on-line documentation

Table 6.1 System requirements on FreeBSD

6.1.1 GUI libraries

LispWorks 6.1 for FreeBSD requires that the X11 release 6 (or higher) is installed.

LispWorks 6.1 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+ 2 (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

32-bit LispWorks requires about 160MB to install, and 64-bit LispWorks needs 180MB. This 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 **`cd`** to the directory containing the product you wish to install.

For LispWorks (32-bit) for FreeBSD the directory is **`x86-freebsd`**.

For LispWorks (64-bit) for FreeBSD the directory is **`amd64-freebsd`**.

Now run the one of following scripts.

Note: 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 6.1 for FreeBSD is supplied as a standard package file to download. A back-up CD-ROM is available.

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.

6.3.2 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 name for the Personal Edition is `lispworks-personal-6.1`.

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 6.1. There may be a downloadable patch bundle which upgrades LispWorks to version 6.1.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 6.1.

6.4.2 Information for Beta testers

Users of LispWorks 6.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 6.1.

See “Uninstalling LispWorks for FreeBSD” on page 50 for instructions.

6.4.3 Installation directories

By default LispWorks is installed in `/usr/local/lib/LispWorks`. A symbolic link to the 32-bit executable is placed in `/usr/local/bin/lispworks-6-1-0-`

x86-freebsd. A symbolic link to the 64-bit executable is placed in **/usr/bin/lispworks-6-1-0-amd64-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 32-bit LispWorks Professional/Enterprise software package file is called

```
lispworks-32bit-6.1.tgz
```

and can be found in the **x86-freebsd** directory of the LispWorks 6.1 download or on the back-up CD-ROM.

The 64-bit LispWorks Enterprise software package file is called

```
lispworks-64bit-6.1.tgz
```

and can be found in the **amd64-freebsd** directory of the LispWorks 6.1 download or on the back-up CD-ROM.

The Personal Edition software package file is called

```
lispworks-personal-32bit-6.1.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-32bit-6.1.tgz
   ```

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

   ```
   pkg_add -p <directory> lispworks-32bit-6.1.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-32bit-6.1.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-6-1-0-x86-freebsd`	Personal Edition
`lispworks-6-1-0-x86-freebsd`	Professional or Enterprise Edition
`lispworks-6-1-0-amd64-freebsd`	64-bit Enterprise Edition

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 101 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-6-1-0-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 & FreeBSD”.

6.8 Printable LispWorks documentation

In a default installation, the **`lib/6-1-0-0/manual/offline`** directory contains PDF format versions of the manuals.

These files are also available at **`www.lispworks.com/documentation/`**.

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

6.9 Uninstalling LispWorks for FreeBSD

To uninstall LispWorks, perform the following steps as root:

1. If patches have been installed, then you will first need to uninstall that package:

   ```
   pkg_delete lispworks-patches-32bit-6.1
   ```

   or

   ```
   pkg_delete lispworks-patches-64bit-6.1
   ```

2. Then uninstall the main software package containing LispWorks 6.1:

   ```
   pkg_delete lispworks-32bit-6.1
   ```

   or

   ```
   pkg_delete lispworks-64bit-6.1
   ```

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 6.1. 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 6.1 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 **`lw61-hp-pa.tar`** and **`lwdoc61-unix.tar`**.

For Sun Sparc (32-bit) you need the files named **`lw61-sparc.tar`** and **`lwdoc61-unix.tar`**.

For Sun Sparc (64-bit) you need the files named **`lw61-sparc64.tar`** and **`lwdoc61-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-6-1-0-hp-pa11` would be a LispWorks 6.1 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/6-1-0-0` 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/6-1-0-0`.

   You can move the LispWorks image wherever you prefer, because the value of `*lispworks-directory*` in the supplied image is the path-name `#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 `*lisp-works-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 6.1.

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/6-1-0-0/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/6-1-0-0/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 93.

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-6-1-0-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 56

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 54.

HTML documentation is installed in the **`lib/6-1-0-0/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/6-1-0-0/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 56.

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 58.

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 file **`lwlicense`** in the following places, in order:

- in the current working directory (folder)
- in the directory containing the LispWorks executable
- in the **`Library/lib/6-1-0-0/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 64.

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`** or **`save-image-with-bundle`**.

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

   ```
   % cd "/Applications/LispWorks 6.1/LispWorks.app/Contents/MacOS"
   ```

   or for the X11/GTK+ LispWorks image:

   ```
   % cd "/Applications/LispWorks 6.1"
   ```

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-6-1-0-macos-universal -build /tmp/save-config.lisp
   ```

   or

   ```
   % ./lispworks-6-1-0-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-6-1-0-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 64 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 6.1/LispWorks.app/Contents/MacOS/lispw
orks-6-1-0-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 6.1/LispWorks.app/Contents/MacOS/lispw
orks-6-1-0-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 74.

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-6-1-0-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-6-1-0-x86-win32.exe -build
C:\temp\save-config.lisp > C:\temp\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 74 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-6-1-0-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-6-1-0-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 6.1 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 & 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 file **`lwlicense`** in the following places, in order:

- in the current working directory
- in the directory containing the LispWorks executable
- in the **`lib/6-1-0-0/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 82.

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-6-1-0-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 82 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-6-1-0-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-6-1-0-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 6.1 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/6-1-0-0/manual/offline/pdf`** (28Mb). You can download these PDF format manuals from **`www.lispworks.com/documentation`**, 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/6-1-0-0/manual/online`** directory of the LispWorks library.

11.2 Software Requirements

The LispWorks 6.1 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 6.1 CD-ROM which contains the LispWorks products you have bought, and how to mount it.

11.3.1 The LispWorks 6.1 CD-ROM

The CD-ROM contains images for LispWorks 6.1 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 88.
2. Extract the product files from the `tar` file containing them. This is described in “Installing LispWorks” on page 89.

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_61/
```

To unmount:

```
umount /cdrom/lw_61/
```

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 6.1.

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 6.1 in the same directory as previous versions of LispWorks such as LispWorks 6.0. This is because all the 6.1 files are stored in a subdirectory called **`lib/6-1-0-0`**.

You must recompile all your code with the LispWorks 6.1 compiler.

Binaries produced by **`compile-file`** in previous versions of LispWorks such as LispWorks 6.0 do not load into a LispWorks 6.1 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_61/`, you would type

```
tar -xof /cdrom/lw_61/lw61-sparc.tar
```

11.4.1.4 HP UX (HP Precision Architecture)

The files you need to unpack for LispWorks on HP UX are `lw61-hp-pa.tar` and `lwdoc61-unix.tar`.

The LispWorks image is

```
./lispworks-6-1-0-hp-pa11
```

11.4.1.5 SPARC Solaris (LispWorks 32-bit)

The files you need to unpack for LispWorks (32-bit) on Solaris are `lw61-sparc.tar` and `lwdoc61-unix.tar`.

The LispWorks image is

```
./lispworks-6-1-0-sparc-solaris
```

11.4.1.6 SPARC Solaris (LispWorks 64-bit)

The files you need to unpack for LispWorks (64-bit) on Solaris are `lw61-sparc64.tar` and `lwdoc61-sparc64.tar`.

The LispWorks image is

```
./lispworks-6-1-0-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/6-1-0-0/config`** (see “If you are using the keyfile system” on page 56.
- **`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/6-1-0-0/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 113.

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-6-1-0-sparc-solaris`** is the LispWorks 6.1 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 90, 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/6-1-0-0/`**.

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 96).
- **`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/6-1-0-0/`**, 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 98.

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-6-1-0-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 91.

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/6-1-0-0/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 Build phase (delivery-time) errors

A common cause of errors seen while building (delivering) an application is running part of the application's runtime initialization, or something else that assumes the application is already running.

One error sometimes seen is **`"Not yet multiprocessing."`** and other likely build phase errors include those arising from code that assumes something about the runtime environment.

Such initializations should be done at the start of the runtime phase, as described in "Separate runtime initializations from the build phase" on page 15.

12.1.4 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.

We recommend that you routinely check the size of your image using **`cl:room`**, whether you see warning messages or not.

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 is close to the limit: it 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. It will usually manage to free the required space, but if it cannot then crashes may result. Therefore you should take action to reduce allocation or increase available memory when you see this message.

Check the size of the image, both by **`cl:room`** and by OS facilities (such as **`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*.

12.1.5 Corrupted LispWorks executable

Programs which attempt to clean up your machine by automatically removing data they identify as unnecessary may accidentally corrupt your LispWorks executable, because they do not understand its format. This will prevent LispWorks from starting.

Examples are the `prelink` cron job on Linux and CleanMyMac on Macintosh. These particular programs should no longer affect LispWorks, but there may be similar utilities in use.

If corruption occurs check if it has been caused by a clean-up utility. If this is the case, firstly configure your clean-up utility to ignore LispWorks, and then reinstall LispWorks.

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 108 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 6.1 supports any Linux distribution with glibc 2.3.3 or 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-6.1-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-6.1-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-6.1-1
#
```

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-6.1-1.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-6.1-1.i386.rpm
```

12.3.3 Using multiple versions of Motif on Linux

The version of Open Motif required by LispWorks 6.1 with the Motif GUI may not be compatible with other applications (including LispWorks 4.2). It is however compatible with LispWorks 6.0, LispWorks 5.x, LispWorks 4.4 and 4.3, so you for example you should be able to run LispWorks 6.1 and LispWorks 6.0 simultaneously with either Open Motif installed.

Whilst it is not supported for LispWorks 5.1 and later versions, 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 6.1 image is actually using, look in the bug form. See “Generate a bug report template” on page 114 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 162 and “Troubleshooting on X11/Motif” on page 108.

12.4.1 GTK+ version

GTK+ 2 (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 108 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 108 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:

```
lwdoc61-unix.tar;1
```

then check the **`mount`** command used (“Mounting the CD-ROM” on page 88).

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/6-1-0-0`) 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 110 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 110 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 110) 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 **`~/Lisp-works`** 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 6.1/lispworks-6-1-0-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 amd64 Solaris)
`patches/system/0001/0001.64ffasl` (for amd64 FreeBSD)
`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.

Private patch loading is controlled by the file:

```
lib/6-1-0-0/private-patches/load.lisp
```

`private-patches/` is the 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 6.1/Library/lib/6-1-0-0/private-
patches/my-patch.nfasl
```

On Windows:

```
<install>\lib\6-1-0-0\private-patches\my-patch.ofasl
```

On Linux:

```
<install>/lib/6-1-0-0/private-patches/my-patch.ufasl
```

On UNIX:

```
<install>/lib/6-1-0-0/private-patches/my-patch.pfasl (for HP-PA)
<install>/lib/6-1-0-0/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
```

as in the commented 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.

Note: On Windows Vista and Windows 7 you should save private patches and edit `private-patches/load.lisp` while logged in as the user who will run LispWorks. If there are multiple LispWorks users, each user should do this.

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 6.1/LispWorks.app/Contents/MacOS/lispw
   orks-6-1-0-macos-universal" -init init.lisp > lw.out
   ```

   where `%` denotes a Unix shell prompt.

   On Windows:

   ```
   C:\> "Program Files\LispWorks\lispworks-6-1-0-x86-
   win32.exe" -init init.lisp > lw.out
   ```

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

   On Linux:

```
% /usr/bin/lispworks-6-1-0-x86-linux -init init.lisp > lw.out
```

where `%` denotes a Unix shell prompt.

On UNIX (SPARC in this example):

```
% /usr/lib/lispworks/lib/6-1-0-0/config/lispworks-6-1-0-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 114 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 64-bit LispWorks for FreeBSD

LispWorks Enterprise (64-bit) for FreeBSD is now available. Upgrade routes are available from LispWorks (32-bit) for FreeBSD.

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 Professional and Enterprise development images need SSE2

LispWorks 6.x Professional and Enterprise Editions require a Pentium 4-compatible processor with SSE2 technology. However, runtimes built with it run on any Pentium-compatible processor

Also, LispWorks 6.x Personal Edition runs on any Pentium-compatible processor.

13.1.4 Older platforms

LispWorks 6.1 (32-bit) for Macintosh supports Mac OS X 10.3, 10.4, 10.5 on PowerPC, and Mac OS X 10.4, 10.5, 10.6 and 10.7 on Intel.

LispWorks 6.1 (64-bit) for Macintosh supports Mac OS X 10.5 on PowerPC, and Mac OS X 10.5, 10.6 and 10.7 on Intel.

13.2 Multiprocessing

LispWorks supports Symmetric Multiprocessing (SMP) on Microsoft Windows, Mac OS X, Linux, FreeBSD and x86/x64 Solaris platforms. Where functionality differs from other platforms, the documentation refers to "SMP LispWorks" or "Non-SMP LispWorks", as appropriate.

The remainder of this section describes changes made since the introduction of SMP LispWorks in version 6.0.

13.2.1 gc-generation replaces mark-and-sweep

`hcl:mark-and-sweep` is deprecated. Please use **`hcl:gc-generation`** instead.

13.2.2 Simple processes not supported

Simple processes are no longer supported.

Old APIs to create them, including **`mp:create-simple-process`** and calling **`mp:process-run-function`** with **`:simple-p t`** now signal error. In particular, **`mp:simple-process-p`** and **`mp:*default-simple-process-priority*`** are no longer documented.

13.2.3 Better ways to use condition variables

New functions **`mp:lock-and-condition-variable-wait`**, **`mp:lock-and-condition-variable-signal`**, **`mp:simple-lock-and-condition-variable-wait`**, and **`mp:lock-and-condition-variable-broadcast`** are now the recommended interfaces.

The **`mp:lock-and-condition-*`** functions perform the equivalent of locking, and then doing something and calling the corresponding **`mp:condition-*`** function within the scope of the lock.

These new functions make it simpler to code, help to avoid mistakes, and optimize some cases. They are now the recommended interface for condition variables.

13.2.4 New function for peeking a mailbox

The new function **`mp:mailbox-not-empty-p`** tests whether a mailbox has contents. It is faster than **`mp:mailbox-peek`**.

13.2.5 Mailbox names for easier debugging

`mp:make-mailbox` now has a *name* argument which can be very useful for debugging. The name appears in the printed representation of the mailbox, and also in the state of any process that waits for it.

13.2.6 New function barrier-block-and-wait

`mp:barrier-block-and-wait` enables a barrier and then waits until a specified number of arrivers arrive. It eliminates the need for **`mp:process-wait`**, which may cause delays due to delays in running the scheduler.

13.2.7 New function processes-count

`mp:processes-count` returns the number of Lisp processes that are currently alive.

13.2.8 More places for which low-level atomic operations are defined

Low-level atomic operations such as **`atomic-push`** and **`compare-and-swap`** implicitly ensure order of memory between operations in different threads. These are now supported for places as follows, additionally to the places documented in LispWorks 6.0:

(**`the`** *type place*)	For a *place* which is itself supported
(**`symbol-value`** *symbol*)	

This was documented (but not actually implemented) in LispWorks 6.0.

symbol

This was implemented (but not documented) in LispWorks 6.0.

For the details see the section "Low level atomic operations" in the "Multiprocessing" chapter of the *LispWorks User Guide and Reference Manual*.

13.2.9 Internal server processes

Some processes are now marked as "internal server" which means they respond to requests for work from other processes. The main effect of this is that if the only processes that are remain are "internal servers", nothing is going to happen, so LispWorks quits.

For more information see the function **`mp:process-run-function`** and the new functions **`mp:process-internal-server-p`** and **`mp:any-other-non-internal-server-process-p`**.

Note: On Windows, **`com:server-can-exit-p`** tests for internal servers.

13.3 GTK+ window system

LispWorks 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+ 2 (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 162.

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 6.1 with Motif you also need Imlib installed, as described earlier in this manual.

13.4 New CAPI features

See the *CAPI Reference Manual* for more details of these, unless directed otherwise.

13.4.1 Drag'n'drop images

On Cocoa and GTK+ you can now drag and drop images, by using the **`:image`** format with a **`gp:image`** object. See the new section "Drag and Drop" in the *CAPI User Guide*.

13.4.2 Drag'n'drop improvements on GTK+

On GTK+, **`capi:list-panel`** and **`capi:tree-view`** now support dropping, via *drop-callback*.

Also dragging and dropping a list of URIs is now supported on GTK+. See the discussion in **`capi:set-drop-object-supported-formats`**.

13.4.3 Drag'n'drop improvements on Cocoa

On Cocoa, **`capi:list-panel`** and **`capi:tree-view`** now support dropping, via *drop-callback*.

13.4.4 New class browses and displays HTML on Windows and Cocoa

`capi:browser-pane` embeds a pane that displays HTML, using WebKit on Cocoa and IWebBrowser2 on Microsoft Windows.

The API allows you to navigate, refresh, handle errors and redirect to another URL programmatically.

13.4.5 Support for multiple monitors

CAPI now supports positioning (and querying the position of) windows on multiple monitors.

New functions **capi:screen-monitor-geometries**, **capi:screen-internal-geometries** and **capi:pane-screen-internal-geometry** support the notions of monitor geometry (which includes "system areas" such as the Mac OS X menu bar and the Microsoft Windows task bar) and internal geometry (which excludes the system areas).

There is a "primary monitor" which displays any system areas. The origin of the coordinate system (as returned by **capi:top-level-interface-geometry** and **capi:screen-internal-geometry**) is now the topmost/leftmost visible pixel of the primary monitor. Thus the origin may be within a system area such as the Mac OS X menu bar.

The new function **capi:virtual-screen-geometry** returns a rectangle just covering the full area of all the monitors associated with a screen.

Note that code which relies on the position of a window should not assume that a window is located where it has just been programmatically displayed, but should query the current position. This is because the geometry includes system areas where CAPI windows cannot be displayed.

Note also that CAPI does not currently support multiple desktops, which are called workspaces in Linux distros, and called Spaces on Mac OS X.

13.4.6 Initial geometry constraints

The new **capi:element** initarg **:initial-constraints** allows you to supply geometry hints that apply during creation but not after display.

13.4.7 New support for input methods

capi:output-pane and **capi:editor-pane** now support "composition", meaning composing input characters into other characters by an input method, which is needed for input of non-alphabetic languages with a Western keyboard.

See the **capi:output-pane** initarg **:composition-callback** in the *CAPI Reference Manual*.

13.4.8 Initarg for control of automatic resize

`capi:pinboard-object` and **`capi:simple-pane`** have a new initarg **`:automatic-resize`** which controls how the object's geometry behaves when it is inside a **`capi:static-layout`** which changes geometry. The effect is as for **`capi:set-object-automatic-resize`**, but the new initarg allows you to set this up at **`make-instance`** time.

13.4.9 Setting the unsaved document indicator on Cocoa

The new function **`capi:interface-document-modified-p`** gets and sets the document-modified flag in a **`capi:interface`**. Currently this only has a visible effect on Cocoa, where an interface whose document is modified is flagged by adding a dark dot in the middle of its Close button (the red button at top-left of the window).

13.4.10 Interacting with a title bar pathname on Cocoa

The new **`capi:interface`** initarg **`:pathname`** allows you to specify the interface pathname. Cocoa allows dragging this pathname from the title bar, and also provides a context menu on the title bar allowing the user to select any component of the pathname and display it in the Finder.

13.4.11 Specifying the drag image

A **`capi:simple-pane`**'s *drag-callback* can now be made to specify the image that is used in the dragging in most circumstances.

Also a top-level interface can now specify a drag image, which on Cocoa is a 16x16px image to the left of the window title. See the new **`capi:interface`** initarg **`:drag-image`** and also the description of **`capi:simple-pane`**'s *drag-callback*.

13.4.12 More support for images in choices

`capi:list-panel` now supports images via the initargs **`:image-function`**, **`:image-lists`**, **`:image-width`** and so on. The images appear to the left of each item. State images are supported on Microsoft Windows, GTK+ and

Motif. **Note:** This functionality was actually added in LispWorks 6.0 but was omitted from the documentation.

`capi:tree-view` now supports standard images specified by keywords such as **`:std-cut`** and **`:std-file-open`**.

Additionally on Windows, **`capi:tree-view`**, **`capi:list-panel`** and **`capi:toolbar`** all now support view images specified by keywords such as **`:view-large-icons`** and **`:view-sort-date`**, and history images specified by keywords such as **`:hist-back`** and **`:hist-favorites`**.

13.4.13 Keyboard searching in a list-panel

The new **`capi:list-panel`** initarg **`:keyboard-search-callback`** allows you to provide a function that is used to search for and select an item in the list when the user types ordinary characters.

13.4.14 Recent items in text-input-pane on Cocoa

On Cocoa **`capi:text-input-pane`** now supports a recent items menu using **`NSSearchField`**. See the new initargs **`:search-field`** and **`:recent-items`**.

13.4.15 Change callback in text-input-range

`capi:text-input-range` now supports a callback which is called when the user edits the text in the pane. See the new **`capi:text-input-range`** initarg **`:change-callback`**.

13.4.16 Controlling the selection in a display-pane

You can now control the selection in a **`capi:display-pane`** programmatically. See **`capi:display-pane-selection`** and related functions.

13.4.17 Selection functions in toolbar-component

`capi:toolbar-component` now supports **`:selection-function`**, **`:selected-item-function`** and **`:selected-items-function`** initargs like **`capi:menu-component`**. For an example, see **`examples/capi/elements/toolbar.lisp`**.

13.4.18 Tab images in tab-layout on Windows

On Microsoft Windows `capi:tab-layout` now supports per-tab images via the new initargs `:image-function` and `:image-lists`.

13.4.19 ICO image format supported in some situations on Windows

On Microsoft Windows `capi:button` now supports images loaded from `.ico` files by `gp:load-image` or `gp:read-external-image`. There is also limited support for such images by `gp:draw-image`.

This new approach preserves transparency, unlike `gp:load-icon-image`.

13.4.20 Alternative images for button now work on Windows

`capi:button` initargs *disabled-image*, *armed-image* and *selected-disabled-image* now work on Microsoft Windows provided you are running with the themed look-and-feel (which is the default).

13.4.21 image-list in option-pane

Images used in `capi:option-pane` can now be associated with a `capi:image-list` by using the new `:image-lists` initarg.

13.4.22 option-pane with correct theme on Windows 7 and Vista

Some of the features of `capi:option-pane` cause it to be drawn without the UI theme in Microsoft Windows Vista and Windows 7. There is a new initarg `:window-styles` which overcomes this problem when its value includes the keyword `:simple-text-only`.

13.4.23 Image count in make-scaled-general-image-set

The new argument *image-count* in `capi:make-scaled-general-image-set` allows you to specify the number of sub-images.

13.4.24 Color in list items on Cocoa

`capi:list-panel` now supports color in the list items on Cocoa.

To use this you supply a suitable value for the `:color-function` initarg.

LispWorks 6.0 and earlier versions support `:color-function` on some other window systems.

13.4.25 Alternating background color in lists and trees

The new initarg `:alternating-background` allows you to influence the use of alternating background color in `capi:list-panel` and `capi:tree-view` on Cocoa and GTK+.

13.4.26 Default initial width in a multi column list

In a `capi:multi-column-list-panel` you can now specify the default initial width of a column, which the user can resize. To do this, supply `:default-width` in the *columns* specification.

13.4.27 list-view always highlights the selection

`capi:list-view` now always highlights the selected item. In previous releases, it was only highlighted when the pane had the focus.

13.4.28 Automatically arranged icons in a list-view

The new `capi:list-view` initarg `:auto-arrange-icons` determines whether icons are automatically arranged to fit the size of the window.

13.4.29 Controlling slider display

The new initarg `:tick-frequency` allows you to specify whether a `capi:slider` has tick marks, and how the tick marks are spaced. In LispWorks 6.0 and earlier versions, slider tick marks do not appear on all platforms, and there is no control over them in CAPI.

On Microsoft Windows and GTK+, the new initarg `:print-function` and accessor `capi:slider-print-function` allow you to control how the slider displays its current value.

13.4.30 Slider value on Microsoft Windows

On Windows **`capi:slider`** now responds to the **`:show-value-p`** initarg. The value is displayed in a tooltip that is visible only while the user moves the slider with a mouse.

13.4.31 Controlling the action buttons in prompt-with-list

`capi:prompt-with-list` now has arguments allowing control over which "action buttons" are displayed, and their callbacks. Action buttons act on the choice in the dialog (as distinct from "dialog buttons" which dismiss the dialog).

The action buttons typically include **All** and **None**, to select or deselect all of the choice items, but you can modify this.

13.4.32 Alternative action callback

In a **`capi:choice`**, the new *alternative-action-callback* is invoked by a gesture which is the *action-callback* gesture modified by the **`Shift`** key on Windows and GTK+, and modified by the **`Command`** key on Cocoa.

13.4.33 text-input-choice can be read-only on Microsoft Windows

`capi:text-input-choice` now implements *enabled* **`:read-only`** on Windows. In LispWorks 6.0 and earlier versions *enabled* **`:read-only`** does not work on Windows.

enabled **`:read-only`** still works on non-Windows platforms.

13.4.34 Text background in labelled-line-pinboard-object

The new **`:text-background`** initarg for **`capi:labelled-line-pinboard-object`** allows you to control the background color of line labels.

13.4.35 Cocoa application message :finished-launching

The new message callback **`:finished-launching`** for a **`capi:cocoa-default-application-interface`** is invoked just after the user has started the application and all other initialization has been done. You can use it to open a default document for example.

13.4.36 Controlling the input method on GTK+

On GTK+ you can now control whether `capi:output-pane` and `capi:editor-pane` use the native input method to interpret keyboard input.

See the new initarg `:use-native-input-method` which controls this per instance, and also the function `capi:set-default-use-native-input-method` which controls the default behavior.

13.4.37 Tooltips on GTK+

`capi:display-tooltip` is now implemented for GTK+ versions 2.12 and later, though the `:x` and `:y` keywords might not be handled.

13.4.38 Querying the state of a sorted-object

The new function `capi:sorted-object-sorted-by` returns the current sorting type and reverse flag for a `capi:sorted-object`.

13.4.39 Controlling copying when the active pane is a choice

You can now control what `capi:active-pane-copy` will copy when the active pane is a `capi:choice`. This is useful if the choice items are not the same as what the user actually sees in the choice.

To do this, define methods for `capi:item-pane-interface-copy-object` whiich is called by the method of `capi:pane-interface-copy-object` that specializes on choice.

13.4.40 apply-in-pane-process-if-alive

The new function `capi:apply-in-pane-process-if-alive` is like `capi:apply-in-pane-process`, except that it calls its *function* argument only if the pane is alive. A pane is alive when it has a `capi:interface` and the interface has a representation on the screen.

13.4.41 Metafiles supported on GTK+

Metafiles are now supported on GTK+ where Cairo is supported (GTK+ version 2.8 and later). The external metafile format is PDF.

13.4.42 GDI+ metafiles on Windows

On Microsoft Windows, CAPI metafiles now support GDI+ drawing. They also support a combined GDI and GDI+ format. Use this "dual" format if the metafile may be used in a context where only GDI (and not GDI+) is available, but you want the quality of GDI+ where possible.

See the *format* values **`:enhanced`**, **`:enhanced-plus`** and **`:enhanced-gdi`** in **`capi:with-external-metafile`** and **`capi:with-internal-metafile`**.

13.4.43 Native printing on GTK+

Printing using Cairo is now supported, where Cairo is supported (GTK+ version 2.8 and later).

In LispWorks 6.0 only Postscript printing was supported on GTK+.

13.4.44 Predicate for availability of metafiles

The new function **`capi:can-use-metafile-p`** returns true if metafiles can be used.

13.4.45 Metafile example

This new example shows how to draw a rotating metafile:

```
examples/capi/graphics/metafile-rotation.lisp
```

13.4.46 Efficient redisplay of menu bar

`capi:redisplay-menu-bar` has a new keyword argument **`:redo-items`**.

The value *redo-items* controls whether items in a **`capi:menu`** or **`capi:menu-component`** that have an *items-function* get recomputed when **`capi:redisplay-menu-bar`** is called. It defaults to **`t`** in order to ensure that any accelerators are up-to-date.

If the menu bar does contain menus (including sub-menus and menu-components) that have an *items-function*, then if you call **`capi:redisplay-menu-bar`** often (for example, each time the user types a character) it is better to pass *redo-items* **`nil`**.

13.4.47 :documentation option in define-interface

The **`:documentation`** option for pane, layout and menu descriptions in **`capi:define-interface`** is now passed through to the **`defclass`** to provide the documentation of the slot in the class.

13.5 New graphics ports features

For details see the Graphics Ports chapters in the *CAPI Reference Manual* and the *CAPI User Guide*.

13.5.1 Quality drawing

Graphics Ports now supports quality drawing, as well as LispWorks 6.0 style drawing in case your program still needs that.

13.5.1.1 Specifying the drawing mode

There are two possible values for a port's *drawing-mode*:

- **`:quality`** causes all drawing operations to make full use of the graphics port *transform*. In particular, text, images and ellipses are scaled and rotated and coordinates are treated as floating point rather than being rounded to the nearest integer. It also supports anti-aliasing and alpha blending. *operation* is not supported in this mode, but *compositing-mode* is supported (see below).

  On Microsoft Windows **`:quality`** uses GDI+.

  On GTK+ **`:quality`** uses Cairo, which is available in GTK+ version 2.8 and later.
- **`:compatible`** generates the same output as previous versions of Lisp-Works. Text and images are not scaled or rotated, ellipses are not rotated

correctly and other shapes are drawn as if the drawing function was called with transformed coordinates

In most cases the drawing mode defaults to `:quality` so the output of your program may differ from LispWorks 6.0.

Note: `gp:pixblt`, `gp:clear-rectangle` and the graphics state parameters *mask-x* and *mask-y* are all deprecated. For alternatives see “copy-area added, pixblt deprecated” on page 139 and the sections immediately following it.

To use the old drawing mode, do for example:

```
(make-instance 'capi:output-pane :drawing-mode :compatible ...)
```

For more information see "Drawing mode and anti-aliasing" in the *CAPI User Guide*.

13.5.1.2 Querying the drawing mode

The new function `gp:port-drawing-mode-quality-p` tests whether a port does `:quality` drawing.

A graphics port does quality drawing if it was not made with *drawing-mode* `:compatible`, and the underlying library supports this. Microsoft Windows and Cocoa always support it, GTK+ supports it from version 2.8 onwards, and Motif never supports it.

`gp:port-drawing-mode-quality-p` is used in this example:

```
examples/capi/graphics/images-with-alpha.lisp.
```

13.5.2 Anti-aliased text and drawing

The new graphics state parameter *text-mode* controls the mode of rendering text, most importantly anti-aliasing. To draw text without anti-aliasing use:

```
(gp:draw-string pane string x y :text-mode :plain)
```

To draw text the way it would be drawn if *drawing-mode* was `:compatible`, use:

```
(gp:draw-string pane string x y :text-mode :compatible)
```

The new graphics state parameter *shape-mode* controls the mode of drawing shapes, that is anything except text. For example to draw a rectangle without anti-aliasing use:

```
(gp:draw-rectangle pane x y w h :shape-mode :plain)
```

Currently *text-mode* and *shape-mode* are supported only when the *drawing-mode* is `:quality`, described in "Quality drawing" on page 134. For further details see the manual page for `gp:graphics-state`.

13.5.3 Control over combining of drawn graphics

The new graphics state parameter *compositing-mode* controls the way that new drawings are combined with the existing value in the target of the drawing to generate the result.

Two values for *compositing-mode* are supported on Microsoft Windows, Cocoa and GTK+.

- `:over` means draw over the existing values, blending alpha values if they exist.
- `:copy` means that the source is written to the destination ignoring the existing values. If the source has alpha and the target does not, that has the effect of converting semi-transparent source to solid. `:copy` is especially useful for creating transparent and semi-transparent pixmap ports, which can be displayed directly or converted to images by `gp:make-image-from-port`.

A number of other *compositing-mode* values are supported on later versions of Cocoa and GTK+.

compositing-mode is only supported in with *drawing-mode* `:quality`.

For further details and examples see the manual page for `gp:graphics-state`.

13.5.4 Path operations

The new function `gp:draw-path` allows you to draw complex figures, in outline or filled, as specified by a series of drawing operations that include lines, arcs and Bézier curves. A path can also be used as the clipping mask.

`gp:draw-path` works only with *drawing-mode* `:quality`.

13.5.5 Converting between more image formats

Images can be now saved as JPG, PNG, TIFF and other platform-specific formats, as well as BMP.

See `gp:externalize-and-write-image` and `gp:list-known-image-formats`.

13.5.6 apply-rotation-around-point

The new function `gp:apply-rotation-around-point` modifies a transform by a specified rotation around a specified point.

13.5.7 Control over use of transforms

The new macro `gp:with-graphics-transform-reset` allows you to run graphics ports code with any pre-existing transforms ignored within its scope.

The new macro `gp:with-graphics-post-translation` allows you to run graphics ports code with a specified translation that is applied after all pre-existing transforms.

13.6 New color system features

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

13.6.1 Creating color specs with alpha

The new function `color:color-with-alpha` returns a color spec based on the supplied color and with the supplied alpha component.

13.7 Other CAPI and Graphics Ports changes

13.7.1 Origin of coordinates of interfaces

The coordinates of interfaces are now always with respect to the top-left corner of the primary monitor. That includes the result of **`capi:top-level-interface-geometry`**, the **`:x`** and **`:y`** arguments to **`capi:set-top-level-interface-geometry`**, and the initialization arguments **`:x`**, **`:y`**, **`:best-x`** and **`:best-y`**.

In LispWorks 6.0 and previous versions these values were with respect to the top-left of the "work-area" of the screen, which excludes "system areas" like taskbars and the menu bar on Cocoa. This made it difficult to position interfaces consistently. There were also various bugs causing inconsistent behavior.

The effect is most visible on Cocoa, where the vertical origin is now at the top of the Mac OS X menu bar rather than at the bottom of it.

The values returned by **`capi:screen-internal-geometry`** are now also always measured from the top-left of the monitor. That is a change only on Cocoa, where (in LispWorks 6.0 and earlier versions) the result for the primary monitor was measured from the bottom of the Mac OS X menu bar.

We use the term *screen rectangle* to mean a rectangle (*x y width height*) where (*x y*) are coordinates relative to the top-left of the primary monitor. The full area of the primary monitor, including any system areas, is the *primary screen rectangle*. For information about the primary monitor, see “Support for multiple monitors” on page 125.

13.7.2 Window sizes on Cocoa no longer include the interface toolbar

The CAPI functions to return or set the size of a top level interface (for example **`capi:top-level-interface-geometry`**) no longer include the interface toolbar in that size on the native Mac OS X GUI. This has been changed because the toolbar can be resized (or hidden completely) independently of the rest of the window, so including the toolbar’s size leads to unrepeatable calculations.

13.7.3 Default font type on Windows

On Microsoft Windows, font lookups by default now look only for TrueType fonts. This includes **`gp:find-best-font`** and users of it such as the font in **`capi:simple-pane`**.

To get the same behavior as LispWorks 6.0 use

```
    (gp:make-font-description ... :type :wild)
```

Note that the default *drawing-mode* **`:quality`** (see “Quality drawing” on page 134) supports only TrueType fonts. Non-TrueType fonts are supported only with *drawing-mode* **`:compatible`**.

13.7.4 Graphics state object documented

`gp:graphics-state` is now exported, and the manual page contains the detailed description of the various graphics state parameters such as *font*, *background*, *transform* etc.

13.7.5 More font functions and types documented

The types **`gp:font`** and **`gp:font-description`** are now documented. A **`gp:font`** object corresponds to a font in the native system, while a **`gp:font-description`** object is the normal way of specifying a font in CAPI.

The predicates **`gp:font-single-width-p`** and **`gp:font-dual-width-p`** are now documented.

13.7.6 copy-area added, pixblt deprecated

The function **`gp:pixblt`** is deprecated. This is because *drawing-mode* **`:quality`** does not support the *operation* argument and also because **`gp:pixblt`** ignores transforms which means it does not always work as expected. In particular, it can it can draw to the wrong coordinates inside the *display-callback* of **`capi:output-pane`**.

Use instead the new function **`gp:copy-area`**, which copies a rectangular area from one graphics port to another taking account of the transform.

See also the **`gp:graphics-state`** parameter *compositing-mode* for a way to control how **`gp:copy-area`** blends the source and the target.

13.7.7 clear-rectangle deprecated

The function **gp:clear-rectangle** is deprecated. This is because it ignores the graphics state, which means it does not work properly with other drawings. In particular it does not work properly in the *display-callback* of **capi:output-pane**.

Instead call:

```
(gp:draw-rectangle ... :filled t :foreground color
                   :compositing-mode :copy :shape-mode :plain)
```

13.7.8 mask-x and mask-y deprecated

Graphics state parameters *mask-x* and *mask-y* are deprecated. With *drawing-mode* **:quality**, you can now include a translation in *mask-transform*, if required. With *drawing-mode* **:compatible**, *mask-x* and *mask-y* still work on GTK+ and X11 / Motif (as in LispWorks 6.0 and earlier).

13.7.9 Using integers as colors is not allowed on Windows

LispWorks 6.0 for Windows (and earlier versions) allowed integers to be used as the *foreground* and *background* graphics state parameters in Graphics Ports drawing functions such as **gp:draw-string** and **gp:draw-rectangle**. This was not documented and should not be used. In particular, it will not work with quality drawing.

13.7.10 rich-text-pane selection deprecated

The **:selection** argument of **capi:rich-text-pane-character-format** and **capi:set-rich-text-pane-character-format** is deprecated, and if *selection* is **nil** an error is now signalled, since we believe the old functionality was not useful. Please contact Lisp Support if this causes a problem for you.

13.7.11 Destroying a Cocoa application interface

On Cocoa, **capi:destroy** can now remove an individual application interface when running in the LispWorks IDE.

13.7.12 Cocoa application interface example improved

The example in **`examples/capi/applications/`** illustrating the use of an application interface in CAPI on Cocoa has been improved and split into two files for clarity:

`cocoa-application-single-window.lisp` shows how you can control the application menu and a single document window.

`cocoa-application.lisp` shows how you can control multiple document windows and the Dock menu, in addition to the application menu.

13.7.13 delete-item-callback in extended-selection-tree-view

The **`:delete-item-callback`** initarg of **`capi:tree-view`** is now documented, and its interface has changed.

In LispWorks 6.0 and earlier *delete-item-callback* makes multiple calls, one with each selected item. In LispWorks 6.1 and later it makes a single call with a list of the selected items. Existing programs that use **`:delete-item-callback`** in the subclass **`capi:extended-selection-tree-view`** need to be changed. The example in **`examples/capi/choice/extended-selection-tree-view.lisp`** shows a program that works both before and after this change.

13.7.14 Home and End in text-input-pane

A bug whereby the **Home** and **End** keys would invoke in-place completion in a **`capi:text-input-pane`** has been fixed.

Now, **Home** and **End** move to the start and end of the pane, whilst **Up** and **Down** still invoke in-place completion.

13.7.15 titled-menu-object fix

A **`capi:titled-menu-object`** now calls its *title-function* correctly on its interface, as documented.

13.7.16 Initforms fix

A slot in a CAPI interface definition can now have an initform that instantiates a pane, for example

```
    :initform (make-instance 'capi:text-input-range
                             :title "Title" :callback 'my-callback)
```

In LispWorks 6.0 and earlier an error would be signalled when displaying such an interface.

13.7.17 Pixmap ports inherit background etc from their pane

`gp:create-pixmap-port` and `gp:with-pixmap-graphics-port` now default the background and foreground of the pixmap to the values in their *pane* argument.

In LispWorks 6.0 and earlier versions, these values default to those in the owner of pane instead.

For an example illustrating these different behaviors, see the manual page for `gp:with-pixmap-graphics-port`.

13.7.18 Port graphics state accessor functions

The functions `gp:graphics-port-font`, `gp:graphics-port-transform`, `gp:graphics-port-foreground` and `gp:graphics-port-background`, which access slots in the `gp:graphics-state` of a graphics port, are now documented.

In LispWorks 6.0 and earlier, these functions were undocumented and were implemented as macros.

13.7.19 Image Access dimension readers

The functions `gp:image-access-height` and `gp:image-access-width` are now documented. These functions get the dimensions of the image underlying an Image Access object.

13.7.20 pi-related constants documented

The constants `gp:2pi`, `gp:pi-by-2`, `gp:fpi`, `gp:f2pi` and `gp:fpi-by-2` are now documented.

13.8 More new features

For details of these, see the documentation in the *LispWorks User Guide and Reference Manual*, unless a manual is referenced explicitly.

13.8.1 Environment access API

LispWorks now has various environment access APIs which are based on those specified by *Common Lisp: the Language* (2nd Edition):

`hcl:augment-environment`, **`hcl:variable-information`**, **`hcl:function-information`** and **`hcl:declaration-information`**.

There is also an additional function **`system:map-environment`**.

13.8.2 IPv6

Internet Protocol Version 6 (IPv6) is now supported in the socket stream interface.

A new type **`comm:ipv6-address`** represents IPv6 addresses which can now be returned by **`comm:get-host-entry`**.

Functions already accepting an IPv4 address argument, such as **`comm:open-tcp-stream`**, **`comm:start-up-server`**, **`comm:ip-address-string`** and so on have been extended to accept a **`comm:ipv6-address`** instead.

The scope ID is supported and a new function **`comm:parse-ipv6-address`** converts a string to a **`comm:ipv6-address`**.

13.8.3 Terminating a server process

The new function **`comm:server-terminate`** terminates a server process (as started by **`comm:start-up-server`**) and frees all the associated resources.

In LispWorks 6.0 and earlier versions, **`mp:process-kill`** is the way to terminate servers. This is deprecated, because it may leave some value in an invalid state.

13.8.4 Subclassing socket-stream

The new function **`comm:connect-to-tcp-server`**.attempts to connect to a socket on a server and returns a handle for the connection if successful. The handle can then be used as the *socket* when making a **`comm:socket-stream`**.

In most situations you would use instead **`comm:open-tcp-stream`** which connects and returns a **`comm:socket-stream`** object. **`comm:connect-to-tcp-server`** is useful when you want to subclass **`comm:socket-stream`**, passing the handle as the *socket* when instantiating your subclass.

13.8.5 ASDF 2 included

The LispWorks distribution now includes ASDF 2. To use ASDF in LispWorks:

1. Do

   ```
   (require :asdf)
   ```

   This loads the distributed version of ASDF 2, and then the **`asdf-integration.lisp`** example file is loaded automatically. This makes the LispWorks IDE support ASDF, in addition to LispWorks' own **`lw:defsystem`** implementation.
2. Configure ASDF as described in its documentation.
3. Optionally, if you want your later version of ASDF, do

   ```
   (asdf:load-system :asdf)
   ```
4. Load your ASDF system definitions.

   Now you are all set to work with ASDF systems in the Editor, Search Files and System Browser tools.

Alternatively, to use a specific version of ASDF, instead of steps 1-4 you should load it directly and then call **`(provide "asdf")`** to prevent the distributed version from being loaded.

The automatic loading of **`asdf-integration.lisp`** is controlled by the new variable **`lispworks:*autoload-asdf-integration*`**.

13.8.6 Improved interface to temp files

The new functions **`hcl:open-temp-file`** and **`hcl:create-temp-file`** allow you to create a temporary file which is guaranteed to be new.

The undocumented function **`sys:make-temp-file`** still exists as in previous LispWorks versions, but is now deprecated. If you use it, please change to **`hcl:open-temp-file`** or **`hcl:create-temp-file`**.

The new function **`sys:set-temp-directory`** allows you to set the default directory used for temp files.

13.8.7 New function hash-table-weak-kind

`hcl:hash-table-weak-kind` returns the weakness state of a hash table.

13.8.8 Predicate for single-thread arrays

The new function **`hcl:array-single-thread-p`** is the predicate for whether an array is one know to be only accessed in a single thread context, as created by **`(make-array ... :single-thread t)`** or set by **`hcl:set-array-single-thread-p`**.

13.8.9 Copy or append a file

The new functions **`lw:copy-file`** and **`lw:append-file`** copy or append the contents of a file to another file.

13.8.10 Very large heap now supported on Linux

64-bit LispWorks for Linux now automatically adjusts its default heap size on startup, depending the age of the operating system.

On old Linux systems the address space is limited to 512Gb and LispWorks 6.0 reflected this, with the heap by default limited to around 160Gb. Newer Linux systems have a much larger address space and LispWorks 6.1 exploits this by adjusting its range to 4Tb.

For more details see the "Heap size" section in the "64-bit LispWorks" chapter of the *LispWorks User Guide and Reference Manual*.

13.8.11 Killling a pipe's process

The new function **`sys:pipe-kill-process`** tries to kill the process of a pipe stream.

13.8.12 Named pipes on Windows

On Microsoft Windows there are now functions interfacing to named pipes, which is a useful inter-process communication mechanism. See **`win32:open-named-pipe-stream`** (for the server side) and **`win32:connect-to-named-pipe`** (for the client side).

13.8.13 Impersonation on Windows

The new macros **`win32:impersonating-user`** and **`win32:impersonating-named-pipe-client`** allow you to execute code while impersonating a specified user or the current user.

This can be useful in a service process which has sufficient privileges, to perform specific operations with the credentials of some user rather than the system user, or to start an interactive process running as the current user.

13.8.14 Locating the Program Files and Start menu folders

`sys:get-folder-path` now supports the keywords **`:program-files`**, **`:programs`** and **`:common-programs`** on Microsoft Windows.

13.8.15 Hiding the DOS window when calling system commands

On Microsoft Windows, **`sys:call-system`**, **`sys:call-system-showing-output`** and **`sys:open-pipe`** with a string *command* argument now explicitly hide the DOS window containing the console that **`cmd.exe`** starts. You can override this if required.

13.8.16 :use-pty argument in open-pipe

The new keyword argument *use-pty* in **`sys:open-pipe`** is useful on Unix-like systems if the sub-process behaves differently when running interactively and non-interactively.

13.8.17 Test for displayed CAPI windows

The new function `hcl:any-capi-window-displayed-p` is a predicate for whether any CAPI window (other than dialogs) is currently displayed.

13.9 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.

13.9.1 Help menu improvements

Help > Manuals now raises a submenu accessing the LispWorks manuals in HTML format. This menu is easier to use than the modal dialog seen in LispWorks 6.0 and earlier versions.

On Microsoft Windows the Google Chrome browser is now supported.

13.9.2 Tool pre-selected in Preferences

The Preferences window now appears with the appropriate tool pre-selected, when possible.

Raise the Preferences window by **Tools > Preferences...**. On Cocoa, press `Command+,` (comma) or select **LispWorks > Preferences...**.

13.9.3 Graph preferences pre-selected

When you invoke the Preferences window via the context menu in a graph such as the Class Browser's Subclasses view, the Preferences window now appears with the appropriate tool and tab pre-selected.

13.9.4 "...Alt is Meta key" preference applies to all tools

On Microsoft Windows the preference option

Environment > Emulation > Keys > Editor keys like Emacs, Alt is Meta key

now applies to all IDE tools, so your `Alt` key behaves consistently.

In previous LispWorks for Windows releases, the Editor, Output, Listener, Debugger, Profiler and Shell tools use it, but other tools do not.

There is no corresponding change in behavior on other LispWorks platforms.

13.9.5 New preference "Use separate Editor windows for each file"

A new preference option makes the LispWorks Editor tool behave more like typical Mac OS X editors by using a separate window for each file. To change this setting select:

Environment > General > Window Options > Use separate Editor windows for each file

When this is checked, LispWorks will open a separate Editor window for each file (or editor buffer) that you have in memory. In addition, closing an Editor window will remove the underlying editor buffer from memory, possibly asking if you want to save it. The default is unchecked.

13.9.6 Triple-click line selection

The Editor now responds to the triple-click mouse gesture by selecting the current line, in common with typical editor applications.

13.9.7 Better placement of the Find dialog

The **Edit > Find/Replace** dialog now defaults to a position relative to its owner (typically the current Editor tool) rather than at top-left of the screen.

Also the dialog now reappears at its previous position when it is invoked again.

13.9.8 Compilation Conditions Browser tabs and preferences

The **Conditions** tab of the Compilation Conditions Browser has been converted to two tabs called **All Conditions** and **Errors**, which makes it easy to view the errors only.

Also the **Types to Display** preference has been removed.

13.9.9 Notifier buttons changed on Windows

On Microsoft Windows, the buttons in the Notifier window (which appears after an error is signalled) have been re-arranged: **OK** is now immediately to the left of **Abort**.

13.9.10 Editor tool solely as buffers list

You can now use an Editor tool solely as a buffers list. To do this, set it to be non-reusable by switching off the option **Tools > Customize > Reuse Editor**, and select the **Buffers** tab.

13.9.11 Controlling the style of incremental search highlight

You can now specify the style used to highlight matches during incremental searches in the Editor tool.

By default the inverse style is used. To change that, visit **Preferences... > Environment > Styles > Colors and Attributes**, select **Style Name** = **Search Match** and choose colors and attributes.

13.9.12 Searching buffers

You can now use the Search Files tool to search in all currently open editor buffers. To do this, select **Opened Buffers** in the dropdown list in the toolbar.

13.9.13 Searching very large files

The Search Files tool now generates a helpful message when a file is skipped because it is very large. In LispWorks 6.0 and earlier, such files are quietly ignored.

You can still configure the threshold at which the tool skips large files using the **Maximum file size to search** preference.

13.9.14 Better handling of errors in display callbacks

Errors in CAPI display callbacks and focus-related callbacks tend to be invoked repeatedly. Therefore display of panes where such an error occurs ("broken" panes) is now normally disabled until the Debugger tools exists.

Therefore while debugging such errors some panes will not be displayed correctly.

Exiting the Debugger tool will allow the error to recur, so to prevent this in some cases there is now a restart to disable the display of the broken pane permanently.

13.9.15 Improved Saved Sessions GUI

The Saved Sessions controller is now an ordinary window rather than a dialog. Changes you make here are effected immediately.

13.9.16 Layout Dividers in the Interface Builder

It is now possible to add a divider to a row or column layout by using the **Divider** button in the **Basic Panes** tab. The Interface Builder can now parse and generate `capi:define-interface` forms containing `:divider` in a layout description.

13.10 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.

13.10.1 Window splitting commands

The new editor commands `Split Window Horizontally` (`Ctrl+X 5`) and `Split Window Vertically` (`Ctrl+X 6`) split the current window.

`Unsplit Window` (`Ctrl+X 7`) removes the split, or with a prefix argument removes all splits.

13.10.2 Positioning of incremental search matches

A new editor variable `editor:incremental-search-minimium-visible-lines` controls positioning of the matches during an incremental search. The value determines the minimum of visible lines between the match and the top or bottom of the window.

You can display and set the value of editor variables with the commands `Show Variable` and `Set Variable`.

13.10.3 Fixed line position mode for incremental search

The commands `Incremental Search` and `Reverse Incremental Search` (`Ctrl+S` and `Ctrl+R` in Emacs emulation) now have a "fixed line position" mode, which can be invoked by supplying the line number as the prefix argument.

For example, to display successive definitions one line from the top of the text view of the Editor window, do

```
Ctrl+U 2 Ctrl+S ( d e f Ctrl+S Ctrl+S
```

13.10.4 New command deleting up to a specified character

The new editor command `Zap To Char` (`Meta+Z`) is similar to the GNU Emacs command zap-to-char. It prompts for a character and kills text from the current point to the next occurrence of that character.

13.10.5 Displaying operator and highlighted arguments in echo area

If the command `Function Arglist Displayer` is passed a prefix argument, for example by ``Ctrl+U Ctrl+` `` then it does not raise a displayer window, but instead displays the operator and its arguments, with highlight, in the Echo Area. This display is interface-specific, and implemented only for the Editor and other tools based on the editor.

13.11 Foreign Language interface changes

See the *LispWorks Foreign Language Interface User Guide and Reference Manual* for details of these changes.

13.11.1 Embedded dynamic modules

You can now incorporate a dynamic module into a LispWorks image. The data is stored inside the Lisp image and the module can be loaded and used at run time. These are called embedded dynamic modules.

The formats supported are a DLL on Microsoft Windows, a Mach-O dynamically-linked shared library on Mac OS X, and a shared object or shared library on other platforms.

Use an embedded dynamic module when you want to integrate foreign code, and the foreign code is not expected to be available on end-users' computers. In principle this could also be achieved by supplying the foreign module as a separate file together with the LispWorks image, locating it at run time and loading it with **`fli:register-module`**. The embedded dynamic modules mechanism simplifies this.

See "Incorporating a foreign module into a LispWorks image" in the *LispWorks Foreign Language Interface User Guide and Reference Manual* for an overview of embedded dynamic modules.

13.11.2 Foreign blocks

LispWorks for Macintosh now supports foreign blocks: objects that correspond to the opaque "Block" object in C and derived languages that are introduced in CLANG and used by Apple Computer, Inc.

13.11.3 Controlling allocation of foreign objects

A previously undocumented keyword argument to **`fli:allocate-foreign-object`** has been changed. The old argument **`:allocator`** is deprecated, and **`:allocation`** is now supported and documented. This is consistent with other functions which take an **`:allocation`** argument.

The value **`:static`** causes memory to be allocated in the C heap. It must be explicitly freed once the object is no longer needed. The value **`:dynamic`** allocates memory within the scope of the body of **`fli:with-dynamic-foreign-objects`**.

13.11.4 Dynamic library delivery: new example on Linux

There is a new example which creates a LispWorks dynamic library and also a test program for loading it on Linux. See the 4 files in the LispWorks library at

```
examples/delivery/dynamic-library/linux/
```

To build and run the example, follow the instructions in `rundll.c`.

13.11.5 process-foreign-code no longer supported

The function `fli:process-foreign-code` is no longer supported.

As a replacement, the C code can be placed in a file and added to a `lw:defsystem` definition with the member options `:type :c-file`.

This change only affects 32-bit LispWorks on Solaris and HP-UX. (These are the only platforms that supported `fli:process-foreign-code` in LispWorks 6.0 and previous versions).

13.12 COM/Automation changes

This section applies only to Microsoft Windows platforms. See the *LispWorks COM/Automation User Guide and Reference Manual* for details.

13.12.1 Running an executable Automation server

COM now has several functions which simplify running an executable Automation server. In typical cases, you can simply use `com:automation-server-main` as the *function* argument in `deliver` or the *restart-function* argument in `save-image`. For more complicated cases, `com:automation-server-main` takes keywords that modify its behavior, and the underlying functionality is also available and can be used directly: `com:automation-server-top-loop`, `com:set-automation-server-exit-delay` and `com:automation-server-command-line-action`.

Also there are two new predicates for testing whether a COM server is in use or can exit: `com:server-in-use-p` and `com:server-can-exit-p`.

13.12.2 Server can receive requests from more than one application

The new option `:coclass-reusable-p` for `com:define-automation-component` allows you to specify that the server running the component can receive requests from more than one application.

13.12.3 New Automation active object APIs

If other applications need to be able to find one of your running objects from its coclass, then call `com:register-active-object` to register an interface pointer for the object in the system Running Object Table. Use `com:get-active-object` to look for a registered running instance, and call `com:revoke-active-object` to remove the registration.

13.12.4 co-create-guid

The new function `com:co-create-guid` makes a new unique `com:refguid` object, optionally updating the table of known refguids.

13.12.5 get-object

The function `com:get-object` returns an interface pointer for a named object.

Note: This function was documented in LispWorks 6.0, but was omitted from the implementation until the LispWorks 6.0.1 patch release.

13.12.6 Handling Automation exceptions

The new condition class `com:com-dispatch-invoke-exception-error` is used by the LispWorks COM API when Automation signals an exception (`DISP_E_EXCEPTION`). The new function `com:com-dispatch-invoke-exception-error-info` retrieves information stored in such an error.

13.13 Objective-C changes

13.13.1 New function objc-class-method-signature

The new function `objc:objc-class-method-signature` tries to find the relevant method (given an Objective-C class and method name) and returns its signature.

The new types `objc:objc-unknown`, `objc:objc-c++-bool` and `objc:objc-at-question-mark` may appear in the result of `objc:objc-class-method-signature`.

13.14 Common SQL changes

13.14.1 Prepared statements

The SQL expression passed to the functions **`sql:execute-command`**, **`sql:query`**, **`sql:do-query`**, **`sql:simple-do-query`**, **`sql:map-query`** and the **`loop for...being each`** record syntax can now be a prepared statement object created by the new function **`sql:prepare-statement`**.

The Common SQL square bracket reader syntax has been extended to support numbered bind-variables using the form **`[`*`n`*`]`** where *n* is a positive integer.

13.14.2 MySQL library location on Windows

The way that you configure LispWorks for the location of the MySQL library on Microsoft Windows has been changed to make it more flexible when 32-bit and 64-bit programs co-exist on a machine. See the new variable **`sql:*mysql-library-sub-directories*`** and note the changed initial value of **`sql:*mysql-library-directories*`**.

13.14.3 encoding for ODBC and PostgreSQL connections

Connections to ODBC now support the **`:unicode`** external format.

Connections to PostgreSQL now support various external formats. The *encoding* argument of **`sql:connect`** also allows you to set the character set in the connection.

See the manual entry for **`sql:connect`** for the details.

13.14.4 PostgreSQL standard_conforming_strings

LispWorks now sets the PostgreSQL session variable **`standard_conforming_strings`** to **`on`** in order to match the escaping that Common SQL uses. Note that this variable is only available in PostgreSQL 8.2 and later, so escaping will not work correctly in older versions.

13.15 Application delivery changes

See the *LispWorks Delivery User Guide* for more details of the changes mentioned in this section.

13.15.1 Runtimes on old x86 CPUs

By default Delivery on x86 platforms 32-bit LispWorks now creates runtimes which will run on any Pentium-compatible CPU.

In the original LispWorks 6.0 release, SSE2 instructions cause exceptions when LispWorks or a runtime runs on an old CPU lacking these relatively new instructions. The restriction still applies to LispWorks Professional and LispWorks Enterprise editions, but the behavior of their runtimes is now controlled by the delivery keyword argument **`:old-cpu-compatible`**.

The restriction does not not apply to LispWorks Personal Edition, which is configured to run on any Pentium-compatible CPU.

13.15.2 Building a PowerPC (or universal) binary not supported on Mac OS X 10.7

LispWorks cannot build PowerPC (or universal) binaries on Mac OS X 10.7 (Lion) because Apple no longer support Rosetta on that version of the operating system. This is a change in the operating system rather than in LispWorks itself.

13.15.3 Delivery now checks for patch loading

`deliver` now checks that **`load-all-patches`** has been called. If **`load-all-patches`** has not been called in the session, then **`deliver`** signals an error.

13.15.4 set-quit-when-no-windows has changed

The semantics of the function **`lw:set-quit-when-no-windows`** have changed slightly in LispWorks 6.1. It is now documented, in the *LispWorks User Guide and Reference Manual*

In LispWorks 6.0 **`lw:set-quit-when-no-windows`** was undocumented and its semantics depended on the original value of the **`:quit-when-no-windows`** delivery keyword.

13.15.5 :versioninfo extensions

On Microsoft Windows, the **`deliver`** keyword **`:versioninfo`** now allows the value **`:default`**, meaning that the version information in the **`:exe-file`** is retained (by default, there is no version info).

Also, the keys **`:string-file-info`** and **`:additional-pairs`** are now accepted in a plist value of **`:versioninfo`**, allowing you to supply a "StringFileInfo" block in the version information and specify non-standard values in that block, such as "MIMEType".

Lastly, the key **`:character-set`** is now documented for **`:versioninfo`**.

13.15.6 :quit-when-no-windows with Automation servers

On Microsoft Windows, if the application is an automation server the checking routine set up by the delivery keyword **`:quit-when-no-windows`** now checks the server (as well as CAPI windows). It does not quit if the server appears to be in use.

See **`com:automation-server-top-loop`** and **`com:set-automation-server-exit-delay`** for more information.

13.15.7 copy-file change may affect Mac OS X application bundle code

A function **`copy-file`** is no longer defined in the example file **`examples/configuration/macos-application-bundle.lisp`**.

LispWorks now includes **`lw:copy-file`** (see “Copy or append a file” on page 145) which is used by that example code.

Therefore if you use code based on an older version of **`macos-application-bundle.lisp`** you should remove the definition of **`copy-file`** from it for use with LispWorks 6.1 and later. Alternatively use **`hcl:create-macos-application-bundle`** or **`hcl:save-image-with-bundle`** instead of the example code.

13.16 CLOS/MOP changes

13.16.1 Initarg checking extended

The system-supplied primary methods for **`reinitialize-instance`**, **`update-instance-for-redefined-class`** and **`update-instance-for-different-class`** now check the initargs as specified by ANSI Common Lisp. In previous LispWorks releases, only **`make-instance`** checked them.

The new function **`clos:set-clos-initarg-checking`** controls this checking. Please use it instead of **`clos:set-make-instance-argument-checking`**.

`clos:set-make-instance-argument-checking` now does the same comprehensive checking as **`clos:set-clos-initarg-checking`**, but is deprecated as its name is no longer accurate.

13.16.2 Stricter checking of generic function lambda list syntax

The lambda list of a generic function is now checked for conformance with the Common Lisp specification. In particular, an error is signaled now if the variables are not symbols.

This affects **`defgeneric`**, **`ensure-generic-function`** and calls to **`make-instance`** for objects of type **`standard-generic-function`**.

13.17 Other changes

13.17.1 The -build command line option loads patches

The **`-build`** command line option now calls **`load-all-patches`**.

This change ensures that patches are always loaded before the build script.

13.17.2 New check for patch loading when saving universal binary

On Mac OS X, **`save-universal-from-script`** now checks that **`load-all-patches`** has been called. If **`load-all-patches`** has not been called in the session, then **`save-universal-from-script`** signals an error.

13.17.3 Using an embedded build script with -build

The **`-build`** command line argument can now read from stdin. This is useful if you want to embed a build script within a shell script that runs LispWorks.

For more information, see the section "Command Line Arguments" in the *LispWorks User Guide and Reference Manual*.

13.17.4 Defining setf expanders in protected packages

The **`defsetf`** or **`define-setf-expander`** forms now check the *access-fn* using the values of **`hcl:*packages-for-warn-on-redefinition*`** and **`lw:*handle-warn-on-redefinition*`**, to guard against setf expanders being defined (or redefined) in protected packages.

13.17.5 Unsetting an environment variable in 64-bit LispWorks

Unsetting a system environment variable now works in 64-bit LispWorks. You would do this by:

```
(setf (lw:environment-variable "ENV_VAR") nil)
```

13.17.6 String reader obeys default character element type

The reader now constructs strings as documented according to the value of **`lw:*default-character-element-type*`**. For example:

```
CL-USER 1 > (set-default-character-element-type 'simple-char)
SIMPLE-CHAR

CL-USER 2 > (type-of "ABC")
SIMPLE-TEXT-STRING
```

In LispWorks 6.0 and earlier versions a bug caused the string reader to make a smaller string where possible (a **`simple-base-string`** in the example above).

13.17.7 Change to package-use-list of COMMON-LISP package

The **`COMMON-LISP`** package now has an empty package-use-list.

In LispWorks 6.0 and earlier the **`COMMON-LISP`** package uses the LispWorks-specific packages **`HCL`** and **`LISPWORKS`**. These have now been removed from the package-use-list. The change is visible mainly when using `cl:apropos` or `lw:regexp-find-symbols` with the `COMMON-LISP` package, for example forms like these no longer find external symbols of the `HCL` and `LISPWORKS` packages:

```
(apropos "COPY" 'common-lisp)

(regexp-find-symbols "COPY-*" :packages (list "CL"))
```

This change also affects `cl:find-symbol` with a name that matches a symbol that is exported by the other packages.

13.17.8 Changes in *features*

`:lispworks6.1` is present, `:lispworks6.0` is not.

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*.

13.17.9 Floating point optimization of +, - and *

Calls to `cl:+` , `cl:-` and `cl:*` with more than 4 arguments are now optimized with the `float 0` and `safety 0` declaration. This improvement was actually made in LispWorks 5.1, however the documentation in LispWorks 6.0 and earlier stated that the optimization would not happen with more than 4 arguments.

13.17.10 Binary file type(s)

The variable `sys:*binary-file-type*` is now documented. It provides the default file type of binary files. `cl:load`, `sys:load-data-file` and `cl:require` recognize this as a binary file type (in addition to `sys:*binary-file-types*`).

The variable `sys:*binary-file-types*` is now exported and documented. It tells `cl:load`, `sys:load-data-file` and `cl:require` about additional file types which should be recognized as binary ("fasl") files. Add a string to the value of `sys:*binary-file-types*` if you want to load binary files which have an extension which is different from the value of `sys:*binary-file-type*`.

13.17.11 Loading old data files

Binary files created with **`hcl:dump-forms-to-file`** or **`hcl:with-output-to-fasl-file`** in LispWorks 6.0, LispWorks 5.x, LispWorks 4.4 or LispWorks 4.3 can be loaded into LispWorks 6.1 using **`sys:load-data-file`**.

13.17.12 ensure-directories-exist with :up

A bug whereby **`cl:ensure-directories-exist`** would signal an error when called with a path containing an **`:up`** component is fixed.

13.17.13 Limiting the printed length of strings

The variable **`hcl:*print-string*`** is now documented. It allows you to specify a maximum length when printing strings.

13.18 Documentation changes

The HTML documention now contains a link to the introduction page, aiding navigation between manuals. Thanks to Rainer Joswig for suggesting this.

Similar manual entries in the *LispWorks User Guide and Reference Manual* are now colaesced into fewer pages, including the INT32 API and the Unicode string comparison functions.

The printable manuals are PDF Version 1.5. You may need to upgrade to a compatible PDF reader.

There is a new section "Guidance for control of the storage management system" in Chapter 10 of the *LispWorks User Guide and Reference Manual*.

Chapter 21 of the *LispWorks User Guide and Reference Manual* now includes an overview of the socket stream interface, and not merely the OpenSSL part of it. This chapter has therefore been renamed "TCP/IP socket communication and SSL".

The *LispWorks Delivery User Guide* chapter "Writing Code Suitable for Delivery" has been substantially revised. In particular there is a new subsection "Error handling in delivered applications" which explains when and how to use **`cl:handler-bind`**, **`cl:handler-case`** or **`cl:*debugger-`**

hook* to handle runtime errors. It also describes how to add logging of errors to your application.

13.19 Known Problems

13.19.1 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.

13.19.2 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.

In GTK+ versions older than 2.12, **capi:display-tooltip** does not work. In version 2.12 and later, the **:x** and **:y** keyword arguments of **capi:display-tooltip** might not be handled.

13.19.3 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.

13.19.4 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.

13.19.5 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.

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).

13.20 Binary Incompatibilty

If you have binaries (fasl files) which were compiled using LispWorks 6.0 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 6.1.

Index