LispWorks®

Release Notes and Installation Guide

Version 7.1

Copyright and Trademarks

LispWorks Release Notes and Installation Guide

Version 7.1

October 2017

Address

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

Telephone

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

From elsewhere: +44 1223 421860

Fax

From North America: 617 812 8283

From elsewhere: +44 870 2206189

www.lispworks.com

Contents

1

Introduction

1.1 LispWorks Editions

LispWorks is available in several product editions on desktop platforms.

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

Note: 32-bit LispWorks on SPARC Solaris is licensed differently to other platforms, as detailed in “32-bit LispWorks for SPARC Solaris” 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. 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.
- HobbyistDV, 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 Hobbyist Edition

LispWorks 7.1 Hobbyist Edition is available to individual licensees for non-commercial and non-academic use. It is a fully-functional Common Lisp IDE without most of the limitations of the Personal Edition:

- No heap size limit.
- No session time limit.
- The functions **save-image** and **load-all-patches** are available.
- Initialization files are available.

HobbyistDV, Professional and Enterprise Edition module loading is not included. In particular, the function **deliver** is omitted so runtimes cannot be generated.

1.1.3 HobbyistDV Edition

LispWorks 7.1 HobbyistDV Edition is available to individual licensees for non-commercial and non-academic use. It has all the features of the Hobbyist Edition plus:

- The function **deliver** allowing generation of non-commercial end-user applications and libraries.

1.1.4 Professional Edition

LispWorks 7.1 Professional Edition includes all the features of the HobbyistDV Edition plus:

- 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

1.1.5 Enterprise Edition

LispWorks 7.1 Enterprise Edition provides further support for the software needs of the modern enterprise. It has all the features of the Professional Edition plus:

- 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. These implementations are licensed separately.

1.1.6 32-bit LispWorks for SPARC Solaris

On SPARC Solaris the Edition model described above does not apply to 32-bit LispWorks. 32-bit LispWorks 7.1 for SPARC Solaris is available with a basic developer license, and the add-on products CLIM, KnowledgeWorks, LispWorks ORB and Application Delivery are each separately available.

64-bit LispWorks Enterprise for SPARC Solaris is separately available and follows the “LispWorks Editions” licensing model described above.

1.2 LispWorks for Mobile Runtime

LispWorks for Android Runtime and LispWorks for iOS Runtime are new products which you can use to build LispWorks runtimes for inclusion in mobile apps.

1.3 Evaluation quick guide

If you are evaluating LispWorks, then the following notes might prove to be useful.

- LispWorks support (**lisp-support@lispworks.com**) will be happy to answer any issues you have.
- The LispWorks distribution contains various examples demonstrating various features of LispWorks. All the examples are in the directory "examples" inside the LispWorks installation.

  You can find this directory by evaluating the following in a LispWorks Listener:

  ```
  (example-file "")
  ```

  Each example contains comments that explain what it demonstrates.

  In many cases it is convenient to copy the example and modify it to do what you want, rather than writing your own code from scratch.
- If you encounter an error that is not obviously a bug in your code, it is always best to produce a full bug report as described in “Generate a bug report template” on page 125. This will speed up the resolution of the issue.
- If you have performance issues, you should use **room**, **extended-time** and **profile** to narrow the problem. See the *LispWorks User Guide and Reference Manual* for details of these diagnostic functions and macros. You should also report it to LispWorks support, as LispWorks is efficient in general and we do not expect performance problems.

1.4 Further details

For further information about LispWorks products visit

www.lispworks.com

To purchase LispWorks please follow the instructions at:

www.lispworks.com/buy

1.5 About this Guide

This document is an installation guide and release notes for LispWorks 7.1 on Mac OS X, Windows, Linux, x86/x64 Solaris, FreeBSD, AIX and SPARC Solaris platforms, and LispWorks for Mobile Runtime. 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.

Unless explicitly mentioned, instructions in this manual refer to the Hobbyist, HobbyistDV, Professional and Enterprise Editions, rather than the Personal Edition or LispWorks for Mobile Runtime which are distributed separately.

1.5.1 Installation and Configuration

Chapters 2-8 explain in brief and sufficient terms how to complete a LispWorks installation on Mac OS X, Windows, Linux, x86/x64 Solaris, FreeBSD, AIX 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”, Chapter 7, “Installation on AIX” or Chapter 8, “Installation on SPARC Solaris”.

Chapter 9 briefly mentions installation of LispWorks for Mobile Runtime.

Chapters 10-13 explain in detail everything necessary to configure, run, and test LispWorks 7.1. Choose the chapter for your platform: Chapter 10, “Configuration on Mac OS X”, Chapter 11, “Configuration on Windows”, Chapter 12, “Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX” or Chapter 13, “Configuration on SPARC Solaris”. 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.

1.5.2 Troubleshooting

Chapter 14, “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.

1.5.3 Release Notes

Chapter 15, “Release Notes” highlights what is new in this release and special issues for your consideration.

2

Installation on Mac OS X

This chapter is an installation guide for LispWorks 7.1 (32-bit) for Macintosh and LispWorks 7.1 (64-bit) for Macintosh. 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.

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 7.1/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 7.1/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 7.1 supports Macintosh computers containing 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 processor. 170MB of disk space including documentation.	Mac OS X version 10.5.x or higher GTK+ 2 (version 2.4 or higher) if you want to run the GTK+ GUI. Open Motif 2.3 and Imlib2 1.4.9 if you want to run the deprecated Motif GUI.

Table 2.1 System requirements on Mac OS X

Product	Hardware Requirements	Software Requirements
LispWorks (64-bit) for Macintosh	Intel processor. 194MB of disk space including documentation	Mac OS X version 10.5.x or higher GTK+ 2 (version 2.4 or higher) if you want to run the GTK+ GUI. Open Motif 2.3 and Imlib2 1.4.9 if you want to run the deprecated Motif GUI.

2.4 Installing LispWorks for Macintosh

2.4.1 Main installation and patches

The LispWorks 7.1 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches.

2.4.2 Information for Beta testers

Users of LispWorks 7.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.1.

See “Uninstalling LispWorks for Macintosh” on page 15 for instructions.

2.4.3 Information for users of previous versions

You can install LispWorks 7.1 in the same location as LispWorks 7.0 or previous versions. If you always choose the default install location, a new folder named **`LispWorks 7.1 (32-bit)`** or **`LispWorks 7.1 (64-bit)`** will be created alongside the other versions.

2.4.4 Launch the LispWorks installer

The LispWorks installer is a **pkg** file, with the following name:

LispWorks71-32bit_Installer.pkg (32-bit LispWorks)

LispWorks71-32bit_Installer.pkg (64-bit Lispworks)

LispWorksPersonal71_Installer.pkg (LispWorks Personal Edition)

To install LispWorks, launch this file, which should run the Mac OS X Installer application. If this does not happen, right-click on th file and choose **Open With > Installer**.

The Introduction page should be displayed. Click **Continue** to go to the next step.

2.4.5 The Read Me

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

2.4.6 The License Agreement

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

2.4.7 Install Location

All the files installed with LispWorks are placed in the LispWorks folder, which is named **LispWorks 7.1 (32-bit)**, **LispWorks 7.1 (64-bit)** or **LispWorks Personal 7.1** depending on which edition you are installing. The LispWorks folder is placed in the main **Applications** folder for use by all users.

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

2.4.8 Choose your installation type

The default Standard Install includes the native Mac OS X GUI and the documentation, but you can also customize the install, for examle to select 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.

2.4.8.1 The native Mac OS X GUI

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

2.4.8.2 The X11 GTK+ and Motif GUIs

If you want to use LispWorks with either of the alternative X11 GUIs, click **Customize** and select the option **LispWorks with X11 IDE** under **Extra items**.

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

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 Imlib2 version 1.4.9 or later.

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

The X11 GUIs are not available for the Personal Edition.

2.4.8.3 The Documentation

If you use the Standard Install the documentation will be installed.

If you do not wish to install the documentation, click **Customize** and uncheck the **LispWorks documentation** option under **Standard items**.

2.4.9 Installing and entering license data

Now click **Install**.

You will be prompted for an administrator's name and password.

If you are not installing the LispWorks Personal Edition, then 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 or Lisp Sales.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, showing the complete output after you enter it, preferably with a screenshot.

2.4.10 LispWorks is added to the Dock

The installer adds LispWorks to the Dock.

2.4.11 Finishing up

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

Note: LispWorks needs to be able find its library at run time 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.

2.4.12 Installing Patches

After completing the main installation of LispWorks, 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.

2.4.13 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).

2.4.14 Obtaining Open Motif and Imlib2

LispWorks 7.1 for Macintosh on X11/Motif requires Open Motif 2.3 and Imlib2 1.4.9.

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

Lisp Support can supply suitable Motif and Imlib2 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 “Install Location” on page 10.

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

If you want to create a LispWorks image that does not start the GUI automatically, then see Section 10.4.5, “Saving a non-windowing image” (this option is not available in the Personal Edition).

See Section 10.3, “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.

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.

Follow this session in the X11 terminal for 32-bit LispWorks (the filenames will be slightly different for 64-bit LispWorks):

```
bash-3.2$ cd "/Applications/LispWorks 7.1 (32-bit)"
bash-3.2$ ./lispworks-7-1-0-x86-darwin-gtk
; Loading text file /Applications/LispWorks 7.1 (32-
bit)/Library/lib/7-1-0-0/private-patches/load.lisp
LispWorks(R): The Common Lisp Programming Environment
Copyright (C) 1987-2017 LispWorks Ltd.  All rights reserved.
Version 7.1.0
Saved by LispWorks as lispworks-7-1-0-x86-darwin-gtk, at 28 Apr
2017 15:05
User lw on machine.lispworks.com
; Loading text file /Applications/LispWorks 7.1 (32-
bit)/Library/lib/7-1-0-0/config/siteinit.lisp
;  Loading text file /Applications/LispWorks 7.1 (32-
bit)/Library/lib/7-1-0-0/private-patches/load.lisp
; Loading text file /u/ldisk/lw/.lispworks
```

The LispWorks GTK+ IDE should appear.

See Section 10.3, “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 and Imlib2 installed, you can use LispWorks with the Motif GUI.

You first must load the Motif GUI into the supplied **`lispworks-7-1-0-x86-darwin-gtk`** or **`lispworks-7-1-0-amd64-darwin-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 the function **env:start-environment**. 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 should run the file **uninstall.command** in the LispWorks folder. This must be run as an administrator user.

2.7 Upgrading the LispWorks Edition

Some LispWorks features such as Delivery, Common SQL and KnowledgeWorks are not available in all Editions. You can add these features by upgrading.

After purchasing your upgrade from **lisp-sales@lispworks.com**, select **Help > Register...** and enter your new license key.

2.8 Upgrading to 64-bit LispWorks

To upgrade from 32-bit to 64-bit LispWorks, contact

lisp-sales@lispworks.com

3

Installation on Windows

This chapter is an installation guide for LispWorks 7.1 (32-bit) for Windows and LispWorks 7.1 (64-bit) for Windows. Chapter 11 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 7 **Start** menu under **Start > All Programs > LispWorks 7.1 > HTML Documentation** or on the Windows 8 start screen.

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 7.1 > 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

The LispWorks 7.1 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches.

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.

3.2.3 Installing over previous versions

You can install LispWorks 7.1 in the same location as LispWorks 7.0, LispWorks 6.x, LispWorks 5.x or LispWorks 4.4.5. This is the default installation location.

You can also install LispWorks 7.1 without uninstalling older versions such as Xanalys LispWorks 4.4 or Xanalys LispWorks 4.3 provided that the chosen installation directory is different.

3.2.4 Information for Beta testers

Users of LispWorks 7.1 Beta should completely uninstall it before installing LispWorks 7.1. 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 **`LispWorks71-32bit.exe`**. You will have downloaded this from the **`x86-win32`** folder.

To install LispWorks (64-bit) for Windows run **`LispWorks71-64bit.exe`**. You will have downloaded this from the **`x64-windows`** folder.

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 or Lisp Sales.

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.

3.2.5.2 Installation location

By default 32-bit LispWorks installs in All Users space in **C:\Program Files (x86)\LispWorks**

By default 64-bit 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.lisp-works.com/documentation**.

3.2.5.4 Installing Patches

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

Patch installation instructions are in the README file accompanying the patch download.

3.2.5.5 Starting LispWorks

After installation LispWorks can be invoked from the Start menu or Start screen (on Windows 8).

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 run time 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. Select **Programs and Features** in the Control Panel or **App & features** in Settings on Windows 10.
2. Select **LispWorks 7.1 (32-bit)** or **LispWorks 7.1 (64-bit)** and click **Uninstall**.

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

3.4 Upgrading the LispWorks Edition

Some LispWorks features such as Delivery, Common SQL and KnowledgeWorks are not available in all Editions. You can add these features by upgrading.

After purchasing your upgrade from `lisp-sales@lispworks.com`, select **Help > Register...** and enter your new license key.

3.5 Upgrading to 64-bit LispWorks

To upgrade from 32-bit to 64-bit LispWorks, contact

lisp-sales@lispworks.com

4

Installation on Linux

This chapter is an installation guide for LispWorks 7.1 (32-bit) for x86/x86_64 Linux, LispWorks 7.1 (64-bit) for x86_64 Linux, LispWorks 7.1 (32-bit) for ARM Linux and LispWorks 7.1 (64-bit) for ARM64 Linux. Chapter 12 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

4.1 Software and hardware requirements

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

Hardware Requirements	Software Requirements
155MB of disk space for Enterprise Edition (32-bit) plus documentation	Any distribution with glibc 2.6 or later for x86/x86_64 and 2.17 or later for ARM/ARM64

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 or 2.3.x and Imlib2 1.4.3 or later 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 7.1 for Linux requires that the X11 release 6 (or higher) is installed. It also requires that either GTK+ or Open Motif with Imlib2 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 2.3 is required to run LispWorks with the Motif GUI.

Download and install Open Motif 2.2.x or 2.3.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 Imlib2 version 1.4.3 or later. Install this from your Linux distribution.

Note: You should be able to run the LispWorks 7.1 Motif GUI and LispWorks 7.0, LispWorks 6.x 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.lisp-works.com/documentation** in any case, and the same manuals are also available there in PostScript format.

4.2 License agreement

Before installing, you must read and agree to the license terms.

To do this download the license script from the link we sent to you.

Now run:

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

or, if you are installing the Personal Edition:

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

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.

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

4.3 Software delivery and installer formats

LispWorks 7.1 for Linux is supplied as a download. Two formats are provided:

- Red Hat Package Management (RPM) files for x86 and x86_64. RPM is a utility like **tar**, except it can actually install products after unpacking them. See Section 4.4.3 for more information
- **tar** files

4.3.1 Contents of the LispWorks distribution

The supplied installers contain all of the relevant modules.

For RPM installations, the RPM package name is **lispworks** (or **lispworks-personal** for the Personal Edition).

The Professional and Enterprise Edition modules are in separately installable RPM packages. These are: CLIM 2.0, KnowledgeWorks, LispWorks ORB, and Common SQL. Section 1.1 provides Edition details.

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.

4.4 Installing LispWorks for Linux

4.4.1 Main installation and patches

The LispWorks 7.1 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches.

4.4.2 Information for Beta testers

Users of LispWorks 7.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.1.

See "Uninstalling LispWorks for Linux" on page 34 for instructions.

4.4.3 Installation from the binary RPM file (x86 and x86_64 only)

For installation on ARM and ARM64, see Section 4.4.4, "Installation from the tar files".

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 7.1 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-7-1-0-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-7-1-0-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 115 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-7.1-`*`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-7.1-*.i386.rpm`** wherever **`lispworks-7.1-*.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. Follow the instructions under Section 4.2, “License agreement”.
2. Locate the RPM installation file **`lispworks-7.1-`*`n`*`.i386.rpm`**.
3. Install or upgrade LispWorks in the standard RPM way, for example:

   `rpm --install lispworks-7.1-`*`n`*`.i386.rpm`

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

   `rpm --install --prefix <`*`directory`*`> lispworks-7.1-`*`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 run time 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 LispWorks Professional and Enterprise Editions only.

File Distribution	Layered Product
`lispworks-clim-7.1-`*`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 main **`lispworks`** package.

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-7.1-`*`n`*`.i386.rpm`	CLIM 2.0

Table 4.3 File distributions for layered products in the Enterprise Edition

File Distribution	Layered Product
lispworks-kw-7.1-*n*.i386.rpm	KnowledgeWorks
lispworks-corba-7.1-*n*.i386.rpm	LispWorks ORB
lispworks-sql-7.1-*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/7-1-0-0/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-7.1-n.i386.rpm
```

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

```
rpm --install --prefix <directory> --replacepkgs
lispworks-7.1-n.i386.rpm
```

4.4.3.7 Installing Patches

After completing the main RPM installation of LispWorks 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 LispWorks are as follows:

Table 4.4 Files for LispWorks

lw71-x86-linux.tar.gz	32-bit LispWorks x86 image, modules and examples
lw71-arm-linux.tar.gz	32-bit LispWorks ARM image, modules and examples
lw71-amd64-linux.tar.gz	64-bit LispWorks x86_64 image, modules and examples
lw71-arm64-linux.tar.gz	64-bit LispWorks ARM64 image, modules and examples
lwdoc71-x86-linux.tar.gz	Documentation in HTML and PDF formats for all architectures

Note: The gzipped files for the LispWorks Personal 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 downloaded files before running the installation script.
3. Run the installation script **lwl-install.sh** (or **lwlper-install.sh** for the Personal Edition). as root if the directory specified by the installation directory requires it (the default does).

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

For example, to install 32-bit LispWorks in **/usr/lispworks**, without documentation you would use:

```
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 LispWorks, 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 try to run LispWorks without a valid key, it prints a message reporting that no valid key was found, and exits.

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 12.2, “License keys”.

4.6 Running LispWorks

In a RPM installation, assuming the default prefix of **/usr**, the LispWorks executable is located in **/usr/lib/LispWorks** or **/usr/lib64/LispWorks** or **/usr/lib/LispWorksPersonal** There is also a symbolic link from the **/usr/bin** directory.

In a **tar** installation, assuming the default prefix of **/usr/local**, the LispWorks executable is located in **/usr/local/lib/LispWorks** or **/usr/local/lib64/LispWorks** or **/usr/local/lib/LispWorksPersonal**.

In both cases, the LispWorks executable should not be moved without being resaved, because it needs to be able to locate the corresponding library directory on startup.

The LispWorks executable is named as shown here:.

lispworks-personal-7-1-0-x86-linux	Personal Edition
lispworks-7-1-0-x86-linux	32-bit LispWorks on x86
lispworks-7-1-0-amd64-linux	64-bit LispWorks on x86_64
lispworks-7-1-0-arm-linux	32-bit LispWorks on ARM
lispworks-7-1-0-arm64-linux	64-bit LispWorks on ARM64

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 111 for details if this does not happen.

4.6.1 Entering the license data

When you run LispWorks for the first time, you will need to enter your license details. This should be done as follows (all on one line) using the appropriate LispWorks executable from the table above (32-bit LispWorks on x86 in this example):

lispworks-7-1-0-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 or Lisp Sales.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, showing the complete output after you enter it.

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

4.7 Configuring the image

You can now configure your LispWorks image to suit your needs and load modules as necessary. For instructions, see Chapter 12, “Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX”.

4.8 Printable LispWorks documentation

In a default installation, the **lib/7-1-0-0/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-7.1
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named **lispworks-patches7.1**. 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 the LispWorks Edition

Some LispWorks features such as Delivery, Common SQL and KnowledgeWorks are not available in all Editions. You can add these features by upgrading.

After purchasing your upgrade from **lisp-sales@lispworks.com**, select **Help > Register...** and enter your new license key.

4.11 Upgrading to 64-bit LispWorks

To upgrade from 32-bit to 64-bit LispWorks, contact

lisp-sales@lispworks.com

5

Installation on x86/x64 Solaris

This chapter is an installation guide for LispWorks 7.1 (32-bit) for x86/x64 Solaris and LispWorks 7.1 (64-bit) for x86/x64 Solaris. Chapter 12 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 OpenSolaris (release 2009.06 or later)

Table 5.1 System requirements on x86/x64 Solaris

Hardware Requirements	Software Requirements
For 64-bit LispWorks, 140MB of disk space	GTK+ 2 (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 7.1 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.lisp-works.com/documentation** in any case, and the same manuals are also available there in PostScript format.

5.2 Software delivery and installer format

LispWorks 7.1 for x86/x64 Solaris is supplied as a standard package file to download.

There are two variants, 32-bit LispWorks and 64-bit LispWorks, so be sure to download the one for which you have purchased a license:

5.2.1 Contents of the LispWorks distribution

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

The package name for 32-bit LispWorks is **LispWorks71-32bit**.

The package name for 64-bit LispWorks is **LispWorks71-64bit**.

5.2.2 Personal Edition distribution

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

The package for the Personal Edition is **LispWorksPersonal71-32bit**.

5.3 Installing LispWorks for x86/x64 Solaris

5.3.1 Main installation and patches

The LispWorks 7.1 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches.

5.3.2 Information for Beta testers

Users of LispWorks 7.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.1.

See "Uninstalling LispWorks for x86/x64 Solaris" on page 43 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-7-1-0-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-7-1-0-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-7-1-0-x86-solaris`**.

Note: LispWorks needs to be able find its library at run time 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 32-bit LispWorks software package file is called **`LispWorks71-32bit`**.

The 64-bit LispWorks software package file is called **`LispWorks71-64bit`**.

The Personal Edition software package file is called **LispWorksPersonal71-32bit**.

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 LispWorks71-32bit all
```

for 32-bit LispWorks, or

```
pkgadd -d LispWorks71-64bit 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 LispWorks, 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 try to run LispWorks without a valid key, it prints a message reporting that no valid key was found, and exits.

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 12.2, "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-7-1-0-x86-solaris	Personal Edition
lispworks-7-1-0-x86-solaris	32-bit LispWorks
lispworks-7-1-0-amd64-solaris	64-bit LispWorks

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 111 for details if this does not happen.

5.5.1 Entering the license data

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

```
lispworks-7-1-0-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 or Lisp Sales.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, showing the complete output after you enter it.

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

5.6 Configuring the image

You can now configure your LispWorks image to suit your needs and load modules as necessary. For instructions, see Chapter 12, “Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX”.

5.7 Printable LispWorks documentation

In a default installation, the **lib/7-1-0-0/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 7.1 have been installed then you will need to uninstall the patch package, by

   ```
   pkgrm -n LispWorksPatches71-32bit
   ```

   or

   ```
   pkgrm -n LispWorksPatches71-64bit
   ```

2. Then uninstall the main software package containing LispWorks 7.1 by executing:

   ```
   pkgrm -n LispWorks71-32bit
   ```

   or

   ```
   pkgrm -n LispWorks71-64bit
   ```

5.9 Upgrading the LispWorks Edition

Some LispWorks features such as Delivery, Common SQL and KnowledgeWorks are not available in all Editions. You can add these features by upgrading.

After purchasing your upgrade from **lisp-sales@lispworks.com**, select **Help > Register...** and enter your new license key.

5.10 Upgrading to 64-bit LispWorks

To upgrade from 32-bit to 64-bit LispWorks, contact

lisp-sales@lispworks.com

6

Installation on FreeBSD

This chapter is an installation guide for LispWorks 7.1 (32-bit) for FreeBSD and LispWorks 7.1 (64-bit) for FreeBSD. Chapter 12 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 32-bit LispWorks plus documentation	FreeBSD 10.x, or later with compat10x (if you want to run LispWorks on older versions of FreeBSD, then please contact Lisp Support)

Table 6.1 System requirements on FreeBSD

Hardware Requirements	Software Requirements
180MB of disk space for 64-bit LispWorks plus documentation	GTK+ 2 (version 2.4 or higher) to run the GTK+ GUI. Open Motif 2.3.x and Imlib2 1.4.9 or later 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 7.1 for FreeBSD requires that the X11 release 6 (or higher) is installed.

LispWorks 7.1 also requires that either GTK+ or Open Motif with Imlib2 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 2.3 is required to run LispWorks with the Motif GUI.

Install Open Motif 2.3.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 Imlib2 version 1.4.9 or later. Install this from the FreeBSD distribution or ports tree.

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.lisp-works.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 download the license script from the link we sent to you.

Now run:

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

or, if you are installing the Personal Edition:

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

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

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

6.3 Software delivery and installer format

LispWorks 7.1 for FreeBSD is supplied as a standard package file (in pkg(8) format) to download.

6.3.1 Contents of the LispWorks distribution

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

The package name for 32-bit LispWorks is `lispworks71-32bit`.

The package name for 64-bit LispWorks is `lispworks71-64bit`.

6.3.2 Personal Edition distribution

You can install the LispWorks Personal Edition by downloading it from `www.lispworks.com/downloads`.

The package name for the Personal Edition is `lispworks71-personal`.

6.4 Installing LispWorks for FreeBSD

6.4.1 Main installation and patches

The LispWorks 7.1 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches.

6.4.2 Information for Beta testers

Users of LispWorks 7.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.1.

See “Uninstalling LispWorks for FreeBSD” on page 51 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-7-1-0-x86-freebsd`. A symbolic link to the 64-bit executable is placed in `/usr/bin/lispworks-7-1-0-amd64-freebsd`.

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 software package file is called

```
lispworks71-32bit-7.1.txz
```

The 64-bit LispWorks software package file is called

```
lispworks71-64bit-7.1.txz
```

The Personal Edition software package file is called

```
lispworks71-personal-7.1.txz
```

6.4.5 Installing LispWorks for FreeBSD

To install LispWorks, perform the following steps as root:

1. Follow the instructions under Section 6.2, “License agreement”.
2. Locate the software package file.
3. Install or upgrade LispWorks in the standard way, for example:

   ```
   pkg_add lispworks71-32bit-7.1.txz
   ```

   This command installs LispWorks in **`/usr/local/lib/LispWorks`**.

Note: LispWorks needs to be able find its library at run time 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 Installing Patches

After completing the main installation of LispWorks, 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 try to run LispWorks without a valid key, it prints a message reporting that no valid key was found, and exits.

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 12.2, “License keys”.

6.6 Running LispWorks

The LispWorks executable is located in the **/usr/local/lib/LispWorks** or **/usr/local/lib/LispWorksPersonal** 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-7-1-0-x86-freebsd	Personal Edition
lispworks-7-1-0-x86-freebsd	32-bit LispWorks
lispworks-7-1-0-amd64-freebsd	64-bit LispWorks

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 111 for details if this does not happen.

6.6.1 Entering the license data

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

lispworks-7-1-0-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 or Lisp Sales.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, showing the complete output after you enter it.

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

6.7 Configuring the image

You can now configure your LispWorks image to suit your needs and load modules as necessary. For instructions, see Chapter 12, "Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX".

6.8 Printable LispWorks documentation

In a default installation, the **lib/7-1-0-0/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 lispworks71-patches-32bit
   ```

   or

   ```
   pkg delete lispworks71-patches-64bit
   ```

2. Then uninstall the main software package containing LispWorks 7.1:

   ```
   pkg delete lispworks71-32bit
   ```

or

```
pkg delete lispworks71-64bit
```

6.10 Upgrading the LispWorks Edition

Some LispWorks features such as Delivery, Common SQL and KnowledgeWorks are not available in all Editions. You can add these features by upgrading.

After purchasing your upgrade from `lisp-sales@lispworks.com`, select **Help > Register...** and enter your new license key.

6.11 Upgrading to 64-bit LispWorks

To upgrade from 32-bit to 64-bit LispWorks, contact

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

7

Installation on AIX

This chapter is an installation guide for LispWorks 7.1 (32-bit) for AIX and LispWorks 7.1 (64-bit) for AIX. Chapter 12 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

7.1 Software and hardware requirements

An overview of system requirements is provided in Table 7.1. The sections that follow discuss any relevant details.

Hardware Requirements	Software Requirements
160MB (180MB) of disk space for 32-bit (64-bit) LispWorks plus documentation	AIX 6.1 or higher.

Table 7.1 System requirements on AIX

Hardware Requirements	Software Requirements
Processor: POWER4 or later.	GTK+ 2 (version 2.4 or higher) to run the GTK+ GUI. Open Motif 2.1.30 and Imlib2 version 1.4.9 or later 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 7.1 System requirements on AIX

7.1.1 GUI libraries

LispWorks 7.1 for AIX requires that the X11 release 6 (or higher) is installed.

LispWorks 7.1 also requires that either GTK+ or Open Motif with Imlib2 are installed.

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

7.1.1.1 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.

7.1.1.2 Motif

Open Motif version 2.1.30 or higher is required to run LispWorks with the Motif GUI.

You will also need Imlib2 version 1.4.9 or later.

7.1.2 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.lisp-works.com/documentation** in any case, and the same manuals are also available there in PostScript format.

7.2 License agreement

Before installing, you must read and agree to the license terms.

To do this download the license script from the link we sent to you.

Now run the following script as root:

```
sh lwa-license.sh
```

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

7.3 Software delivery and installer format

LispWorks 7.1 for AIX is supplied as **tar** files to download. together with two shell scripts which you use at install time.

7.3.1 Contents of the LispWorks distribution

The supplied **tar** file contains all of the relevant modules.

Your license key will control which modules can be used.

7.4 Installing LispWorks for AIX

7.4.1 Main installation and patches

The LispWorks 7.1 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches.

7.4.2 Information for Beta testers

Users of LispWorks 7.1 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.1.

See "Uninstalling LispWorks for AIX" on page 59 for instructions.

7.4.3 Installation directories

By default LispWorks is installed in **`/opt/LispWorks`**. A symbolic link to the 32-bit executable is placed in **`/opt/LispWorks/bin/lispworks-7-1-0-rs6k-aix`**. A symbolic link to the 64-bit executable is placed in **`/opt/LispWorks/bin/lispworks-7-1-0-rs6k64-aix`**.

You can alter the default installation location at install time.

7.4.4 Selecting the correct archives

The 32-bit LispWorks archive is called

```
lw71-rs6k-aix.tar.gz
```

The 64-bit LispWorks archive is called

```
lw71-rs6k64-aix.tar.gz
```

The documentation archive, which contains manuals in HTML and PDF formats, applies to both 32-bit and 64-bit LispWorks:

```
lwdoc71-x86-linux.tar.gz
```

7.4.5 Installing the archive

To install LispWorks, perform the following steps as root:

1. Follow the instructions under Section 7.2, "License agreement".
2. Use **`cd`** to change directory to the location of the tar files before running the installation script.
3. Run the appropriate installation script, either **`lwa-32bit-install.sh`** or **`lwa-64bit-install.sh`** as root.

   This script takes **`--prefix`** and **`--excludedocs`** arguments to control the installation location and amount of documentation installed.

For example, to install 32-bit LispWorks in **/usr/lispworks**, without documentation you would use:

```
sh lwa-32bit-install.sh --excludedocs --prefix /usr/lispworks
```

Note: LispWorks needs to be able find its library at run time 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 7.6 for instructions on entering your license details.

7.4.6 Installing Patches

After completing the main installation, 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.

7.5 LispWorks looks for a license key

If you try to run LispWorks without a valid key, it prints a message reporting that no valid key was found, and exits.

For instructions on entering your license key, see Section 7.6.1, “Entering the license data” below.

For more information about license keys, see Section 12.2, “License keys”.

7.6 Running LispWorks

The LispWorks executable is located in the **/opt/LispWorks/lib/LispWorks-32-bit** or **/opt/LispWorks/lib/LispWorks-64-bit** directory of the installation (assuming the default prefix of **/opt/LispWorks**) 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 **/opt/LispWorks/bin** directory.

The LispWorks executable is named as shown here:.

lispworks-7-1-0-rs6k-aix	32-bit LispWorks
lispworks-7-1-0-rs6k64-aix	64-bit LispWorks

When you run LispWorks, the splashscreen should appear, followed by the LispWorks Podium and a Listener. See “Troubleshooting” on page 111 for details if this does not happen.

7.6.1 Entering the license data

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

```
lispworks-7-1-0-rs6k-aix --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 or Lisp Sales.

If you have problems with your LispWorks license key, send it to **lisp-keys@lispworks.com**, showing the complete output after you enter it.

7.7 Configuring the image

You can now configure your LispWorks image to suit your needs and load modules as necessary. For instructions, see Chapter 12, “Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX”.

7.8 Printable LispWorks documentation

In a default installation, the **lib/7-1-0-0/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.

7.9 Uninstalling LispWorks for AIX

To remove an installation of 32-bit LispWorks in the default location, do this - carefully - as root:

```
rm -rf /opt/LispWorks/lib/LispWorks-32-bit
rm /opt/LispWorks/bin/lispworks-7-1-0-rs6k-aix
```

If you do not also have 64-bit LispWorks installed here, you can also do

```
rm -rf /opt/LispWorks
```

To remove an installation of 64-bit LispWorks in the default location, do this - carefully - as root:

```
rm -rf /opt/LispWorks/lib/LispWorks-64-bit
rm /opt/LispWorks/bin/lispworks-7-1-0-rs6k64-aix
```

If you do not also have 32-bit LispWorks installed here, you can also do

```
rm -rf /opt/LispWorks
```

7.10 Upgrading the LispWorks Edition

Some LispWorks features such as Delivery, Common SQL and KnowledgeWorks are not available in all Editions. You can add these features by upgrading.

After purchasing your upgrade from **lisp-sales@lispworks.com**, select **Help > Register...** and enter your new license key.

7.11 Upgrading to 64-bit LispWorks

To upgrade from 32-bit to 64-bit LispWorks, contact

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

8

Installation on SPARC Solaris

8.1 Introduction

This chapter is a brief installation guide for 32-bit LispWorks 7.1 for SPARC Solaris and 64-bit LispWorks 7.1 for SPARC Solaris. It is not relevant to any other product. Chapter 13 discusses installation and configuration in detail, while this chapter presents the minimum 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 13, “Configuration on SPARC Solaris”.

8.2 Extracting software from the CD-ROM

LispWorks 7.1 for SPARC Solaris is supplied on a CD-ROM.

32-bit LispWorks 7.1 for SPARC Solaris is supplied in a tar archive also containing the associated products CLIM 2.0, KnowledgeWorks, and LispWorks ORB.

64-bit LispWorks 7.1 for SPARC Solaris is supplied in a tar archive containing each of the Editions.

In both cases, additionally there may be a patch installer which upgrades LispWorks to version 7.1.x. You need to complete the main installation before adding patches. You will need root access while installing these products.

8.2.1 Finding out which CD-ROM files you need

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

Platform	Hardware code	OS code
Sun Sparc (32-bit, Solaris 2.8 & later)	**sparc**	**sparc-solaris**
Sun Sparc (64-bit, Solaris 2.8 & later)	**sparc64**	**sparc64-solaris**

Table 8.1 Platforms and associated codes

For Sun Sparc (32-bit) you need the files named **lw71-sparc.tar** and **lwdoc71-unix.tar**.

For Sun Sparc (64-bit) you need the files named **lw71-sparc64.tar** and **lwdoc71-sparc64.tar**.

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

8.2.2 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.

lispworks-7-1-0-sparc-solaris is the 32-bit LispWorks image and **lispworks-7-1-0-sparc64-solaris** is the 64-bit LispWorks image.

8.3 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/7-1-0-0** 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/7-1-0-0**.

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

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

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

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

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

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

   See Section 8.5 below for more information about configuring LispWorks. You will need to install your license key first.

8.4 Obtaining and Installing your license keys

8.4.1 Keyfiles and the license server on SPARC

This section applies to 32-bit LispWorks for SPARC Solaris only. For information about licensing 64-bit LispWorks for SPARC Solaris, see Section 8.4.2 on page 65.

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

8.4.1.1 If you are using the keyfile system

You will need a valid key, placed in a keyfile, for LispWorks to run.

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 14.10, “Reporting bugs” for instructions before sending us a bug report. All contact details are in Section 14.10.8, “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/7-1-0-0/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 8.3 above)

- **lib/7-1-0-0/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 102.

8.4.1.2 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*.

8.4.2 Installing the license key on Sun Sparc (64-bit)

This section applies to 64-bit LispWorks for SPARC Solaris only. For information about licensing 32-bit LispWorks for SPARC Solaris, see Section 8.4.1 on page 64.

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

lispworks-7-1-0-sparc64-solaris --lwlicenseserial *SERIALNUMBER* --lwlicensekey *LICENSEKEY*

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

Your LispWorks license key is supplied on a label on the folder containing the CD-ROM.

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

8.5 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.

8.5.1 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-7-1-0-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 64

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.

8.5.2 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"**.

8.6 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 62.

HTML documentation is installed in the **lib/7-1-0-0/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/7-1-0-0/manual/offline/pdf` subdirectory of the LispWorks library.

8.7 Using Delivery, LispWorks ORB, CLIM 2.0, KnowledgeWorks and Common SQL

These products are licensed differently in 32-bit LispWorks for SPARC Solaris and 64-bit LispWorks for SPARC Solaris.

8.7.1 Using Layered Products in 32-bit LispWorks on SPARC

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 on SPARC” on page 64.

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 66.

Note: There is no additional licensing requirement for Common SQL in 32-bit LispWorks for SPARC Solaris.

8.7.2 Using Layered Products in 64-bit LispWorks on SPARC

To use CLIM 2.0 you need LispWorks Professional or Enterprise Edition.

To use each of LispWorks ORB, KnowledgeWorks and Common SQL you need LispWorks Enterprise Edition.

In all cases you need to load the appropriate module using `require`.

Note: There is no additional licensing requirement for Delivery in 64-bit LispWorks for SPARC Solaris.

9

Installation of LispWorks for Mobile Runtime

This chapter describes installation of LispWorks 7.1 for Android Runtime and LispWorks 7.1 for iOS Runtime.

9.1 Installing LispWorks for Android Runtime

We will send you instructions when you get a license for LispWorks for Android Runtime.

Note: Normally you would first develop and debug your program using LispWorks on a desktop platform, for example LispWorks for Linux. You will then build a runtime library using LispWorks for Android Runtime and incorporate it in an Android project (see "Android interface" in the *LispWorks User Guide and Reference Manual*) before testing it on an Android device.

9.2 Installing LispWorks for iOS Runtime

We will send you instructions when you get a license for LispWorks for iOS Runtime.

Note: Normally you would first develop and debug your program using LispWorks for Macintosh. You will then build a runtime library using LispWorks for iOS Runtime and incorporate it in an Xcode project (see "iOS interface" in

the *LispWorks User Guide and Reference Manual*) before testing it on an iOS device or the iOS Simulator on Mac OS X.

10

Configuration on Mac OS X

10.1 Introduction

This chapter explains how to get LispWorks up and running, having already installed the files 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”

10.2 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/7-1-0-0/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, and LispWorks will exit.

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
- 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.

10.3.2 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**.

10.3.3 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 run time 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 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 74.

10.4.2 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")
```

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

   ```
   % cd "/Applications/LispWorks 7.1 (32-bit)/LispWorks (32-
   bit).app/Contents/MacOS"
   ```

   or for the X11 / GTK+ LispWorks image:

   ```
   % cd "/Applications/LispWorks 7.1 (32-bit)"
   ```

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-7-1-0-x86-darwin -build /tmp/save-config.lisp
   ```

   or

   ```
   % ./lispworks-7-1-0-x86-darwin-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.

10.4.3 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-7-1-0-x86-darwin -build /tmp/save-config.lisp >
/tmp/output.txt
```

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

10.4.4 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.

10.4.5 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)
```

10.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. The '**~**' 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 7.1 (32-bit)/LispWorks (32-
bit).app/Contents/MacOS/lispworks-7-1-0-x86-darwin" -init
my-lisp-init
```

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

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

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

```
% "/Applications/LispWorks 7.1 (32-bit)/LispWorks (32-
bit).app/Contents/MacOS/lispworks-7-1-0-x86-darwin" -init -
-siteinit -
```

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

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

10.6 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")`.

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 Common SQL

To load Common SQL enter, for example:

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

or

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

Initialize the database type at run time, 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.

10.7.2 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.

10.7.3 Special considerations when using Common SQL

10.7.3.1 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.

10.7.3.2 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 User-

name, 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.

10.7.3.3 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.

10.7.3.4 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).

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.

11

Configuration on Windows

11.1 Introduction

This chapter explains how to get LispWorks up and running, having already installed it 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”

11.2 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.

11.3 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.

11.3.1 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.

11.3.2 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 run time 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 11.4, below, and Section 11.5, “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 11.4, below, and Section 11.5, “Initializing LispWorks” for further details.

11.4 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.

11.4.1 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 84.

11.4.2 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 %PROGRAMFILES%\LispWorks
   ```

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

   ```
   C:\Program Files (x86)\LispWorks>lispworks-7-1-0-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.

11.4.3 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 (x86)\LispWorks>lispworks-7-1-0-x86-win32.exe -
build C:\temp\save-config.lisp > C:\temp\output.txt
```

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

11.4.4 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.

11.4.5 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 84 but pass the **:environment** argument to **save-image**. For example:

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

11.5 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 **cl: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-7-1-0-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-7-1-0-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.

11.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 7.1 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")
```

11.6.1 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*

11.7 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*.

11.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

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

and at run time 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 run time call:

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

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

11.8 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.

11.9 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.

12

Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX

12.1 Introduction

This chapter explains how to get LispWorks up and running on Linux, x86/x64 Solaris, FreeBSD or AIX, having already installed it. If you have not done this, refer to Chapter 4, Installation on Linux, Chapter 5, Installation on x86/x64 Solaris, Chapter 6, Installation on FreeBSD or Chapter 7, Installation on AIX.

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"

12.2 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/7-1-0-0/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, and LispWorks will exit.

12.3 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.

12.3.1 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.

12.3.2 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 run time 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 12.4, below, and Section 12.5, “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 12.4, below, and Section 12.5, "Initializing LispWorks" for further details.

12.4 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.

12.4.1 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 92.

12.4.2 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-7-1-0-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.

12.4.3 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.

12.4.4 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 92 but pass the **:environment** argument to **save-image**. For example:

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

12.5 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-7-1-0-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-7-1-0-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.

12.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 7.1 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")
```

12.6.1 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*

12.7 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*.

12.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

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

and at run time 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 run time call:

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

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

12.8 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.

12.9 Documentation on x86/x64 Solaris, FreeBSD and AIX

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

13

Configuration on SPARC Solaris

13.1 Disk requirements

The LispWorks software requires up to 53MB of disk space, 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/7-1-0-0/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/7-1-0-0/manual/online** directory of the LispWorks library.

13.2 Software Requirements

The LispWorks 7.1 for SPARC Solaris 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.

13.3 The CD-ROM

This section explains the organization of the LispWorks 7.1 CD-ROM which contains the LispWorks products you have bought, and how to mount it.

13.3.1 The LispWorks 7.1 CD-ROM

The CD-ROM contains images for LispWorks 7.1 and associated products on your platform or platforms.

13.3.1.1 CD-ROM format

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

13.3.2 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 98.
2. Extract the product files from the **tar** file containing them. This is described in "Installing LispWorks" on page 99.

13.3.3 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.

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

```
/cdrom/lw_71/
```

To unmount:

```
umount /cdrom/lw_71/
```

When you 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.

13.4 Installing LispWorks

This section explains how to install LispWorks, having already mounted the CD-ROM. If you have not done this, refer to Section 13.3, "The CD-ROM". It also describes how you obtain keys to run LispWorks 7.1.

13.4.1 Unpacking the archive

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.

13.4.1.1 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.

13.4.1.2 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_71/**, you would type

```
tar -xof /cdrom/lw_71/lw71-sparc.tar
```

13.4.1.3 SPARC Solaris (LispWorks 32-bit)

The files you need to unpack for LispWorks (32-bit) on Solaris are **lw71-sparc.tar** and **lwdoc71-sparc.tar**.

The LispWorks image is:

```
./lispworks-7-1-0-sparc-solaris
```

13.4.1.4 SPARC Solaris (LispWorks 64-bit)

The files you need to unpack for LispWorks (64-bit) on Solaris are **lw71-sparc64.tar** and **lwdoc71-sparc64.tar**.

The LispWorks image is:

```
./lispworks-7-1-0-sparc64-solaris
```

13.4.2 Keyfiles and how to obtain them

This section applies only to 32-bit LispWorks for SPARC Solaris.

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.

13.4.2.1 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/7-1-0-0/config** (see "If you are using the keyfile system" on page 64.
- **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/7-1-0-0/config**.

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

13.4.2.2 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.

13.4.2.3 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 124.

Once you have the key, write it into a file in one of the places listed in Section 13.4.2.1, and start up the LispWorks image.

13.4.3 The License Server

This section applies only to LispWorks (32-bit) for SPARC Solaris. There is no license server for LispWorks (64-bit) for SPARC 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.

13.5 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 run time.

13.5.1 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-7-1-0-sparc-solaris** is the 32-bit LispWorks 7.1 image for use on Sun Sparc Solaris machines.

As noted in Section 13.4.1.1 on page 99, 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**.

13.5.2 The LispWorks library

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 the installation directory in a partition with more disk space. The installation directory must contain a subdirectory called **lib/7-1-0-0/**.

Among the directories on this subdirectory are the following:

- **config** — various files that can be adjusted in order to customize the image (see Section 13.7 on page 104).
- **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/7-1-0-0/**, 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**.

13.6 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.

13.7 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.)

13.7.1 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.

13.7.2 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 run time 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 13.7.3 below, and see also Section 13.8, "LispWorks initialization arguments" for further details.

13.7.3 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.

13.7.4 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 107.

13.7.5 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-7-1-0-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 100.

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.

13.7.5.1 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.

13.8 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 file-name:

```
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.

14

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.

14.1 Troubleshooting

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

14.1.1 License key errors

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, FreeBSD and AIX, 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, FreeBSD and AIX, 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.

14.1.2 Failure of the load-on-demand system

Module files are in the modules directory **`lib/7-1-0-0/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.

14.1.3 Build phase (delivery-time) errors

A common cause of errors seen while building (delivering) an application is running part of the application's run time 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 run time environment.

Such initializations should be done at the start of the run time phase, as described in "Separate run time initializations from the build phase" in the *LispWorks Delivery User Guide*.

14.1.4 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 discrepancies, 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*.

14.1.5 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.

14.2 Troubleshooting on Windows

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

14.2.1 Private patches not loaded on Windows 7, 8 & 10

Modify **private-patches\load.lisp** only via the menu command **Help > Install Private Patches...**

If your LispWorks installation is in the **%ProgramFiles%** folder and you edit **private-patches\load.lisp** directly, then Windows starts to use a redirected private copy of **load.lisp**. **Help > Install Private Patches...** will not update this copy, and thus your new patches will not be loaded.

If this occurs, the solution is to delete the redirected copy of **load.lisp** from your user profile space. On Windows 8 the location is like this:

```
C:\Users\lw\AppData\Local\VirtualStore\Program Files
(x86)\LispWorks\lib\7-1-0-0\private-patches\
```

14.3 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 14.8, "Troubleshooting on X11 / Motif" below for issues specific to X11 / Motif.

14.3.1 Uninstall requires administrator on Mac OS X

You must be logged on an as administrator in order to run **uninstall.command** to uninstall LispWorks. This is because it uses the **sudo** command.

14.4 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 118 below for issues specific to X11 / Motif.

14.4.1 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 7.1 supports any Linux distribution with glibc 2.6 or later.

14.4.2 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-7.1-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-7.1-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-7.1-1
#
```

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 automatically to the supplied value. We have seen this bug in RPM version 4.2, as distributed with Red Hat 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-7.1-1.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-7.1-1.i386.rpm
```

14.4.3 Using multiple versions of Motif on Linux

The version of Open Motif required by LispWorks 7.1 with the Motif GUI may not be compatible with other applications (including LispWorks 4.2). It is however compatible with LispWorks 7.0, LispWorks 6.x, LispWorks 5.x, LispWorks 4.4 and 4.3, so you for example you should be able to run LispWorks 7.1 and LispWorks 7.0 simultaneously with either Open Motif installed.

While 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 or 2.3 file **`libXm.so`** then set **`LD_LIBRARY_PATH`** to include <*motiflibdir*>.

Note: to find out which version of libXm your LispWorks 7.1 image is actually using, look in the bug form. See “Generate a bug report template” on page 125 for instructions on generating the bug form.

14.5 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 160 and "Troubleshooting on X11/Motif" on page 118.

14.5.1 GTK+ version

GTK+ 2 (version 2.4 or higher) is required to run the LispWorks image as distributed.

14.6 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 118 below for issues specific to X11/Motif.

14.7 Troubleshooting on SPARC Solaris

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

See also "Troubleshooting on X11/Motif" on page 118 for issues specific to X11/Motif.

14.7.1 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:

```
lwdoc71-unix.tar;1
```

then check the **mount** command used ("Mounting the CD-ROM" on page 98).

14.7.2 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/7-1-0-0`**) 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.

14.8 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.

14.8.1 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.

14.8.2 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 120 for more information.

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

14.8.3 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 120 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 120) to reduce the number of colors LispWorks allocates.

14.8.4 Motif mnemonics and Alt

Mnemonic processing on Motif always uses **`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.)

14.8.5 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 14.8.6):

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

14.8.6 X11/Motif resources

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

14.8.7 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 7.1/lispworks-7-1-0-x86-darwin-gtk
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 2.4.8.2, "The X11 GTK+ and Motif GUIs". Restart X11.app and LispWorks after installation of Motif.

14.9 Updating with patches

We sometimes issue patches for LispWorks by email or download.

14.9.1 Extracting simple patches

Save the email attachment to your disk.

See Section 14.9.3.2, “Private patches” below about location of your private patches.

14.9.2 If you cannot receive email

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

14.9.3 Different types of patch

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

14.9.3.1 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.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.ifasl (for AIX 32-bit)
patches/system/0001/0001.rfasl (for 32-bit ARM Linux, Android and iOS)
patches/system/0001/0001.xcfasl (for 32-bit iOS Simulator)
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/0001/0001.64ifasl (for AIX 64-bit)
patches/system/0001/0001.64rfasl (for 64-bit ARM Linux and iOS)
patches/system/0001/0001.64xcfasl (for 64-bit iOS Simulator)
patches/system/0001/0001.wfasl (for SPARC 32-bit)
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 10.4, “Saving and testing the configured image” (Mac OS X), Section 11.4, “Saving and testing the configured image” (Windows) or Section 12.4, “Saving and testing the configured image” (Linux, x86/x64 Solaris or FreeBSD), or Section 13.7.3, “Saving and testing the configured image” (other UNIX).

14.9.3.2 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/7-1-0-0/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 7.1 (32-bit)/Library/lib/7-1-0-0/private-patches/my-patch.xfasl

On Windows (but see note below about the **Install Private Patches...** command):

<*install*>\lib\7-1-0-0\private-patches\my-patch.ofasl

On Linux:

<*install*>/lib/7-1-0-0/private-patches/my-patch.ufasl

On SPARC:

<*install*>/lib/7-1-0-0/private-patches/my-patch.64wfasl (for 64-bit LispWorks)
<*install*>/lib/7-1-0-0/private-patches/my-patch.wfasl (for 32-bit LispWorks)

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

immediately after 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 10.4, “Saving and testing the configured image” (Mac OS X), Section 11.4, “Saving and testing the configured image” (Windows), Section 12.4, “Saving and testing the configured image” (Linux, x86/x64 Solaris or FreeBSD), or Section 13.7.3, “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, the correct way to install private patches is using the menu item **Help > Install Private Patches...**. Select the private patch file with the

Add button and edit the `private-patches/load.lisp` in the lower pane to include the loading form supplied by Lisp Support immediately after the commented example there. Then click **Save Changes**, which will run a helper application that interacts with the Windows User Access Control mechanism to allow you to write the files into the protected Program Files folder.

14.10 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 14.10.8. Please note that we much prefer email.

14.10.1 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`.

14.10.2 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.

14.10.3 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.

14.10.4 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.

14.10.5 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 7.1 (32-bit)/LispWorks (32-
   bit).app/Contents/MacOS/lispworks-7-1-0-x86-darwin" -init
   init.lisp > lw.out
   ```

   where % denotes a Unix shell prompt.

   On Windows:

   ```
   C:\> "Program Files\LispWorks\lispworks-7-1-0-x86-
   win32.exe" -init init.lisp > lw.out
   ```

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

   On Linux:

   ```
   % /usr/bin/lispworks-7-1-0-x86-linux -init init.lisp > lw.out
   ```

   where % denotes a Unix shell prompt.

   On UNIX (SPARC in this example):

   ```
   % /usr/lib/lispworks/lib/7-1-0-0/config/lispworks-7-1-0-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.

14.10.6 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**.

14.10.7 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.)

14.10.8 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 125 for details. You can always get a backtrace from within the debugger by entering `:bb` at the debugger prompt

14.10.9 Sending large files

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

14.10.10 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`.

14.11 Transferring LispWorks to a different machine

This section lists the steps necessary to transfer LispWorks 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/7-1-0-0/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 differs, 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 may need to remove the original installation. Check your license agreement if you are unsure: the text of the shrinkwrap agreement is in the file **lib/7-1-0-0/license.txt**.

Instructions for uninstalling LispWorks are in the per-platform chapters of this manual:

- “Uninstalling LispWorks for Macintosh” on page 15
- “Uninstalling LispWorks for Windows” on page 20
- “Uninstalling LispWorks for Linux” on page 34
- “Uninstalling LispWorks for x86/x64 Solaris” on page 43
- “Uninstalling LispWorks for FreeBSD” on page 51

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.

15

Release Notes

15.1 Keeping your old LispWorks installation

You can install LispWorks 7.1 in the same directory as previous versions such as LispWorks 7.0. This is because most of the 7.1 files are stored in a subdirectory called `lib/7-1-0-0`.

Binaries produced by `cl:compile-file` in previous versions of LispWorks do not load into a LispWorks 7.1 image. You must recompile all your code with the LispWorks 7.1 compiler.

15.2 Updating your code for LispWorks 7.1

Check through these release notes for things you need to update in code that already works in LispWorks 7.0.

If you are updating code that works only in versions earlier than LispWorks 7.0, then you should also consult earlier release notes, which are available at `www.lispworks.com/documentation`.

15.2.1 Conditionalizing code for different versions of LispWorks

When conditionalizing code for different versions of LispWorks, make your code work in the latest version and then conditionalize with feature

expressions if necessary, depending on which previous versions of LispWorks you want to support.

For example, use **`#-lispworks6`** rather than **`#+lispworks7`**. This makes it more likely that the code will work without changes when LispWorks 8 is released in future.

Use only documented features. For an example see "Conditionalization for LispWorks versions" in the entry for **`cl:*features*`** in the *LispWorks User Guide and Reference Manual*.

15.3 Platform support

15.3.1 Runtimes iOS

LispWorks for iOS Runtime supports 64-bit devices now, using a new garbage collector (the Mobile GC).

The example script run-lw-ios.sh now builds 4 images: 32-bit and 64-bit for each of iOS and the iOS simulator.

15.3.2 AIX/PowerPC implementation supports SMP

LispWorks for AIX/PowerPC now supports Symmetric Multiprocessing (SMP), on both 32-bit and 64-bit.

15.3.3 ARM64 Linux implementation

LispWorks (64-bit) for ARM64 Linux (also known as aarch64) is now available.

15.3.4 FreeBSD 10.x support

LispWorks 7.1 supports FreeBSD 10.x and later and is supplied as a standard package file, in pkg(8) format. Older versions of FreeBSD are not supported.

15.3.5 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.

15.3.6 Code signing LispWorks images

15.3.6.1 Signing of the distributed executable

On Mac OS X, the LispWorks Personal Edition application bundle is signed in the name of LispWorks Ltd.

On Microsoft Windows, the LispWorks Personal Edition executable is signed in the name of LispWorks Ltd.

Other LispWorks editions are not signed, because of the complications around image saving and delivery that this would lead to.

15.3.6.2 Signing your development image

On Microsoft Windows and Mac OS X you can sign a development image saved using **`save-image`** with the **`:split`** argument. On Mac OS X, the **`:split`** argument should have value **`:resources`**.

15.3.6.3 Signing your runtime application

On Microsoft Windows and Mac OS X you can sign a runtime executable or dynamic library which was saved using **`deliver`** with the **`:split`** argument.

15.4 Multiprocessing

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

This section describes changes made in the multiprocessing interface since version 7.0, which are documented in the *LispWorks User Guide and Reference Manual*.

15.4.1 Additional functions for use with mailboxes

The new function **mp:mailbox-send-limited** sends an object to a mailbox as long as the mailbox has not reached a specified maximum size.

The new functions **mp:mailbox-size** and **mp:mailbox-full-p** allow the current size and fullness of a mailbox to be queried.

15.4.2 Additional arguments to mp:process-send

The function **mp:process-send** now has extra keyword arguments **:limit** and **:timeout**, to allow the maximum size of the process's mailbox to be controlled.

In addition, a new keyword argument **:error-if-dead-p** controls what happens if the process is dead and the return value is now a boolean.

15.4.3 Additional functions for use with unlocked queues

The new functions **mp:unlocked-queue-count**, **mp:unlocked-queue-size** allow the current number of objects and maximum size of an unlocked queues to be queried.

The new function **mp:unlocked-queue-peek** allows the first object in an unlocked queue to be retrieved without removing it.

15.4.4 Evaluation environment for initial bindings in foreign threads

The initial value forms in **mp:*process-initial-bindings*** are now evaluated in the dynamic environment of the new process when a foreign thread calls into Lisp. In previous releases, the evaluation occurred in a "no-process" scope, and an error would have entered the debugger in the console without an option to abort.

15.4.5 Newly exported system classes for synchronization objects

The system classes **mp:barrier**, **mp:condition-variable**, **mp:lock**, **mp:mailbox** and **mp:semaphore** are now exported and documented.

15.4.6 Safely using globally accessible data

Documentation has be added to describe how to safely make an object's contents accessible to other threads ("globally accessible").

The new macro **`system:globally-accessible`** allows synchronization to be performed when accessing a globally accessible slot in cases where it is not done by some other means.

See "Making an object's contents accessible to other threads" in the *LispWorks User Guide and Reference Manual* for more details.

15.4.7 Timers can cause themselves to be stopped

If a timer function returns the keyword **`:stop`**, then timer is unscheduled, allowing you to schedule a repeating timer that unschedules itself when some condition is true.

15.5 GTK+ window system

LispWorks uses GTK+ as the default window system for CAPI and the LispWorks IDE on Linux, FreeBSD, AIX 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 does not support GTK+.

A few known problems are documented on “Problems with CAPI on GTK+” on page 160.

15.5.1 Using Motif instead of GTK+

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

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

To use LispWorks 7.1 with Motif you also need Imlib2 (on Linux, FreeBSD, Mac OS X and AIX) or Imlib (on Solaris) installed, as described earlier in this manual.

15.5.2 X11/Motif requires Imlib2 instead of Imlib

LispWorks 7.1 requires Imlib2 1.4.3 or later to use the Motif GUI on Linux, FreeBSD, Mac OS X and AIX. Previous versions of LispWorks required Imlib, which is a different library and is still required on Solaris.

15.6 New CAPI features

See the *CAPI User Guide and Reference Manual* for more details of these, unless directed otherwise. This section is not relevant to LispWorks for Mobile Runtime.

15.6.1 capi:stacked-tree class added

The new class **`capi:stacked-tree`** displays a tree where each node has an associated value, with child nodes that represent a fraction of that value. Each node is displayed as a rectangle whose width corresponds to the value. Child nodes are displayed below the node to make a stack of rectangles.

15.6.2 Customizing graph-pane edge objects

The class **`capi:graph-pane`** now has an initarg **`:edge-pane-function`**, which is a function that is called to create an element for each edge. All previous versions of LispWorks have this initarg too, but it has not been documented until now.

15.6.3 Waiting for a function call to return in a pane's process

The new functions **`capi:apply-in-pane-process-wait-single`** and **`capi:apply-in-pane-process-wait-multiple`** call a function in the process associated with a pane and wait for the values to be returned.

15.6.4 Displaying HTML from a string

The new function **`capi:browser-pane-set-content`** sets the contents of a **`capi:browser-pane`** to a string. This is supported on Windows and Mac OS X.

15.6.5 Set the appearance of panes inside interfaces of a specific type

The new functions **capi:set-interface-pane-name-appearance** and **capi:set-interface-pane-type-appearance** set the appearance (foreground, background, font) of panes inside interfaces of a specific type. This allows customization of an application's fonts and colors without changing every occurrence of the pane in the source code.

15.6.6 Simplified way to update internal scroll parameters

The new function **capi:update-internal-scroll-parameters** updates the internal scroll parameters. It is intended to be used in your *scroll-callback* when using internal scrolling (see **simple-pane** and "output-pane scrolling").

15.6.7 Optional new directory prompter on Windows

The function **capi:prompt-for-directory** now displays a more modern-looking directory prompter when the **"shell-objs"** module has been loaded (not the default).

15.6.8 Allowing a layout to change its background color

The *background* of an instance of **capi:layout** (inherited from **capi:simple-pane**) can now be set to **:background** initially, to make the layout without a background initially but allow it to be changed later.

15.7 New graphics ports features

Unless otherwise stated, for details see the Graphics Ports chapters in the *CAPI User Guide and Reference Manual*. This section is not relevant to LispWorks for Mobile Runtime.

15.7.1 Writing an image to a stream using externalize-and-write-image

The function **gp:externalize-and-write-image** can now be passed an output stream.

15.7.2 Controlling the type of image in externalize-image

The function **gp:externalize-image** now has **:type** and other image formatting keyword arguments (such as **:quality**) which are used to specify the type of external image returned (as in **gp:externalize-and-write-image**).

15.7.3 New function to make a scaled image from part of another image

The new function **gp:make-scaled-sub-image** makes a new image from a scaled part of an existing image.

15.7.4 New functions to draw and measure glyphs on Cocoa

The new functions **gp:draw-glyphs**, **gp:draw-glyph**, **gp:get-glyphs-extent** and **gp:get-glyph-extent** draw and measure glyphs. In general, you should use the functions that work with characters such as **gp:draw-string** instead of these glyph functions.

15.8 Other CAPI and Graphics Ports changes

This section is not relevant to LispWorks for Mobile Runtime.

15.8.1 capi:set-text-input-pane-selection works before capi:display

The function **capi:set-text-input-pane-selection** now works before **capi:display** has been called. In previous releases, the earliest it could be called was from a **:after** on **capi:interface-display**.

15.8.2 Extra options for capi:start-drawing-with-cached-display

The **:automatic-cancel** keyword argument to the function **capi:start-drawing-with-cached-display** can now be a function, which is called after the cached display is canceled.

A additional keyword argument **:resize-automatic-cancel** has been added, which has the same effect as as **:automatic-cancel** but controls what happens when the window is resized rather than when it loses the focus.

15.8.3 capi:interface-customize-toolbar is now implemented on Cocoa

The function **capi:interface-customize-toolbar** now raises the standard customization panel on Cocoa. In previous releases, it did nothing on Cocoa.

15.8.4 pane-can-scroll has been replaced by coordinate-origin

The new initarg **:coordinate-origin** replaces the **:pane-can-scroll** initarg in the class **capi:output-pane**.

:coordinate-origin :scrolled is the same as **:pane-can-scroll nil**. **:coordinate-origin :fixed-graphics** is the same as **:pane-can-scroll t**. There is a new value **:coordinate-origin :fixed** which causes all coordinates to be relative to the visible area.

:pane-can-scroll can still be used, but it is deprecated.

See "output-pane scrolling" in the *CAPI User Guide and Reference Manual* for more details.

15.8.5 Korean input methods

Korean input methods now work correctly.

15.8.6 Graphics Ports drawing functions with scale-thickness

The *scale-thickness* option to the Graphics Ports drawing functions now defaults to **t**, as the documentation has always said. This changed in LispWorks 7.0 but was not documented in the Release Notes.

15.8.7 Clearing graphics port works as documented on Windows

On Windows, the functions **gp:clear-graphics-port** and **gp:clear-rectangle** now to work with transparent colors and ignore the mask as documented. They have always worked correctly on other platforms.

15.8.8 Pixels no longer copied from outside the source port

The functions **gp:copy-pixels** and **gp:copy-area** no longer try to copy pixels from outside the source port's rectangle.

15.8.9 Drawing metafiles to pixmap graphics ports on Cocoa

CAPI metafiles can now be drawn to pixmap graphics ports on Cocoa. Previously, this only worked on other platforms.

15.8.10 scaled-image-set now works with extended-selection-tree-view

Scaled image sets created with **capi:make-scaled-general-image-set** can now be used in the image-list of an **capi:extended-selection-tree-view**.

15.9 More new features

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

15.9.1 Support for remote debugging

Support for remote debugging has been added, which allows you to debug a LispWorks process that is running on one machine using a LispWorks IDE that is running on another machine. It is intended to make it easier to debug applications running on machines that do not have the LispWorks IDE, mainly mobile device applications on iOS and Android, but also applications running on servers where you cannot run the LispWorks IDE.

15.9.2 Support for using Asynchronous I/O with SSL

The Asynchronous I/O API has been extended to support connections using SSL. The existing functions **comm:create-async-io-state-and-connected-tcp-socket** and **comm:accept-tcp-connections-creating-async-io-states** now have keyword arguments such as **:ssl-ctx** and new functions **comm:async-io-state-attach-ssl** and **comm:async-io-state-detach-ssl** have been added.

New functions **comm:async-io-state-ssl-side**, **comm:async-io-state-ssl** and **comm:async-io-state-ctx** allow access to details of the SSL objects from an **comm:async-io-state** object.

15.9.3 Support for OpenSSL 1.1

LispWorks 7.1 supports OpenSSL 1.1 as well as older versions of OpenSSL.

Previous versions of LispWorks will not work with OpenSSL 1.1 because the API has changed.

Note: On Windows, you will need to call **comm:set-ssl-library-path** to set the path when using OpenSSL 1.1. See "How LispWorks locates the OpenSSL libraries" in the *LispWorks User Guide and Reference Manual*.

15.9.4 Support for SNI in socket streams

A new keyword argument **:tlsext-host-name** has been added to various functions in the socket stream API such as **comm:open-tcp-stream**, to allow the SNI extension to be set when making an SSL connection.

15.9.5 Control over handshake time in SSL

Functions that create SSL connections (for example **comm:open-tcp-stream**) can now control when to perform the SSL handshake and how long to wait for it using the new keyword argument **:handshake-timeout**. See "Keyword arguments for use with SSL" in the *LispWorks User Guide and Reference Manual*.

In addition, the new functions **comm:socket-stream-handshake** and **comm:async-io-state-handshake** perform a SSL handshake.

15.9.6 User-defined declaration handlers

The new macro **hcl:define-declaration** defines a declaration handler for code walkers. The implementation of is based on the specification in Common Lisp the Language, 2nd Edition.

The new function **hcl:undefine-declaration** removes declaration handlers.

15.9.7 Tracing and advising subfunctions

It is now possible to trace and "advise" subfunctions that are created by **flet**, **labels** or **lambda**. See **cl:trace**, **lw:defadvice** and "Subfunction dspecs" in the *LispWorks User Guide and Reference Manual*.

15.9.8 Specifying function names using a declaration

The **hcl:lambda-name** declaration can be used to change the name of the surrounding lambda, which is useful for debugging purposes and does not affect the behavior of the program. See "Description of hcl:lambda-name" in the documentation of **cl:declare** in the *LispWorks User Guide and Reference Manual*.

15.9.9 Additional function for use with code coverage

The new function **hcl:map-code-coverage-data** calls a function on each of the files in a **hcl:code-coverage-data** object.

15.9.10 Rotating bits within an integer

The new function **lw:rotate-byte** rotates specified bits within an integer.

15.9.11 reduce-memory implemented for 64-bit LispWorks

The function **hcl:reduce-memory** is now also implemented for 64-bit LispWorks.

15.9.12 New function to return the current function name

The new function **hcl:current-function-name** returns the name of the current function, which is useful for debugging.

15.9.13 New function to close a pipe stream

The new function **system:pipe-close-connection** can be used to close the connection underlying the pipe-stream without closing the stream itself, which allows the exit status of the pipe to be read.

15.9.14 New function to return a date/time string

The new function **hcl:date-string** returns a string representing the date and time (including seconds) from a universal time.

15.9.15 Saving profiler results to a file

The new function **hcl:save-current-profiler-tree** saves the current profiler output to a file. This can be used later by the Profiler tool or can be opened in a text editor.

15.9.16 New arguments to set-up-profiler

hcl:set-up-profiler now has *kw-contexts* and *subfunctions* arguments to control profiling KnowledgeWorks contexts and subfunctions.

15.9.17 New keyword to print timing information when profiling

The function **hcl:start-profiling** now takes a **:time** keyword argument, which causes timing information to be printed by **hcl:stop-profiling**.

15.9.18 New output stream variable in the Java interface

The new variables **lw-ji:*to-java-host-stream*** and **lw-ji:*to-java-host-stream-no-scroll*** are bound to output streams that send anything that is written to them to Java (by calling **lw-ji:send-message-to-java-host**). They can be used anywhere an output stream is needed to make the output go to the Java host.

15.9.19 lw-ji:send-message-to-java-host can add text without a newline

The *where-keyword* argument to the function **lw-ji:send-message-to-java-host** now has addition values **:add-no-scroll** and **:add**, which add text without a newline.

This keyword might also be passed to the *send-message-to-java-host* argument to **lw-ji:init-java-interface**.

The Java method **com.lispworks.Manager.addMessage** also allows the *where* argument to be **ADDMESSAGE_ADD** or **ADDMESSAGE_ADD_NO_SCROLL**.

15.9.20 Change to meaning of append in

com.lispworks.Manager.addMessage

The values of **ADDMESSAGE_APPEND** and **ADDMESSAGE_APPEND_NO_SCROLL** for the *where* argument to the Java method **com.lispworks.Manager.addMessage** now insert a newline after the message. In LispWorks 7.0, the newline was inserted before the message so you may have to modify your code if you rely on that.

This also changes the effect of calling **lw-ji:send-message-to-java-host** on Android with **:append** and **:append-no-scroll**.

15.9.21 Changing the Java interface callbacks

The new function **lw-ji:setup-java-interface-callbacks** can be called after **lw-ji:init-java-interface** was called to change the values of *class-finder*, *java-to-lisp-debugger-hook*, *report-error-to-java-host* or *send-message-to-java-host*. This is useful in the situations where LispWorks performs the call to **lw-ji:init-java-interface**, which happens in Android and in a dynamic library delivered with **lw-ji:setup-deliver-dynamic-library-for-java**.

15.9.22 New Java method to wait for Lisp initialization

The method **waitForInitialization** has been added to the Java class **com.lispworks.LispCalls** to wait for Lisp to become initialized.

15.9.23 New constant to represents Java null values

The new constant **lw-ji:*java-null*** represents a Java null pointer, when this needs to be passed from Lisp to Java.

15.9.24 New length and predicate for simple-int32-vector and simple-int64-vector

The new functions **system:simple-int32-vector-length** and **system:simple-int64-vector-length** return the lengths of **system:simple-int32-vector** and **system:simple-int64-vector** objects.

The new functions **system:simple-int32-vector-p** and **system:simple-int64-vector-p** act as predicates.

15.9.25 Counting occurrences of a regular expression in a string

The new function **lw:count-regexp-occurrences** counts the occurrences of a regular expression in a string.

15.9.26 Trimming whitespace from a string

The new function **hcl:string-trim-whitespace** trims whitespace characters from the beginning and end of a string.

15.9.27 Using color and menus in text output

The new function **hcl:write-string-with-properties** is like **cl:write-string**, but when it writes to an Editor buffer it can add properties to the text it has written. This is especially useful for writing in different colors, bold, italic or underlined.

15.9.28 Pushing an element on the end of a list

The new macros **lw:push-end** and **lw:push-end-new** push an element on the end of a list. These macros were available in all previous versions of LispWorks too, but have not been documented until now.

15.9.29 Serial Port API implemented on non-Windows platforms

The Serial Port API, as described in chapter "The SERIAL-PORT Package" in the *LispWorks User Guide and Reference Manual* has been implemented for non-Windows platforms as well.

15.9.30 Obtaining a dspec for an object

The new function **dspec:object-dspec** returns the dspec for an object that represents some definition, such as a function, method or class.

15.9.31 The debugger now shows unused variables at debug level 3

The debugger now shows unused variables in code that is compiled at debug level 3. In LispWorks 7.0 and earlier versions, these variables were eliminated.

15.9.32 Automatic detection of valid file encodings

The new function **system:specific-valid-file-encoding** can be included in the list **system:*file-encoding-detection-algorithm*** to allow detection of the character encoding in a file, according to a configurable list of encodings in **system:*specific-valid-file-encodings***.

15.9.33 New functions to handle errors while printing

The new functions **hcl:safe-format-to-string**, **hcl:safe-prin1-to-string** and **hcl:safe-princ-to-string** format and print to a string without signaling any errors. These functions are intended to be used in code that handles and reports errors, where it is important to avoid recursive errors.

15.9.34 Creating a volatile registry key

The function **win32:create-registry-key** has a new keyword **:volatile**, which allows a volatile registry key to be created.

15.9.35 Accessing android.os.Build on Android

The new function **hcl:android-build-value** returns the value of a field in the **android.os.Build** Java class on Android.

15.10 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. This section is not relevant to LispWorks for Mobile Runtime.

15.10.1 Support for remote debugging

Support for remote debugging has been added, which allows you to debug a LispWorks process that is running on one machine using a LispWorks IDE that is running on another machine. It is intended to make it easier to debug applications running on machines that do not have the LispWorks IDE,

mainly mobile device applications on iOS and Android, but also applications running on servers where you cannot run the LispWorks IDE.

15.10.2 Profiler layout changes

The **Code To Profile** panel has been redesigned, to increase the area that is dedicated to displaying the results.

Its text pane has been moved to a separate tab.

Its **Symbols...** and **Packages...** buttons have been moved to the profiling parameters, opened using the **Profiler > Set Profiling Parameters** menu item.

Its **Profile** button has been moved to the toolbar and is also available from the **Profiler > Profile the 'Code To Profile'** menu item.

15.10.3 New way to display the Profiler results

There is a new **Stacked Tree** tab in the Profiler, which displays the profiler call tree using a rectangle per function call The width of the rectangle corresponds to the time spent in that call.

15.10.4 New ways to filter the Profiler results

Two new context menu items have been added to Profiler result panels.

Set Function As Root displays a combined tree for all of the calls to the specified function. This is useful when the function is called in many places but you need to know now it behaves in aggregate.

Show Calls to Function [inverted] displays an inverted tree with function at its root, and the children being all of the callers of the function. This is a useful way for exploring why a function seems to be on the stack more than expected.

15.10.5 Storing the Profiler results in a file

The Profiler results can now be saved in a file and can be read from a file.

15.10.6 Importing Profiler results

The Profiler results can now be imported from the last use of **`hcl:stop-profiling`** or **`hcl:profile`**.

15.10.7 Profiling background processes

The Profiler can now collect and display results from background processes (as well as those running within the **Code To Profile** panel) using the **Start Profiling** and **Stop Profiling and Import** toolbar buttons and **Profiler** menu items. You can choose which processes to include.

15.10.8 Improved setting of profiling parameters

A dialog is now used to set profiling parameters, including packages, symbols and the interval. It also controls whether profile the GC, add call counters, show unknown call frames and include KnowledgeWorks forward chaining contexts.

The profiler now defaults to profiling all packages.

15.10.9 Syntax coloring in the Listener

The Listener now supports syntax coloring for input.

15.10.10 Protection from deletion of prompts

Text editing commands in the Listener and Echo Area now perform additional checks to try to prevent accidental deletion of the text in the prompt.

15.10.11 Position of the point after double-click in the Editor

Double-clicking in the Editor now leaves the point at the end of the region rather than the start. This is for compatibility with GNU Emacs.

15.10.12 Customizing text and background colors

The text and background colors of various types of text editing pane can now be changed using the **Main Colors** section of the **Styles** tab in the **Environment**

page of the Preference dialog. You can also apply more detailed customization using the new functions **capi:set-interface-pane-name-appearance** and **capi:set-interface-pane-type-appearance**.

15.10.13 New multiple-click-drag behavior in the Editor

Double-clicking in the Editor and then dragging without releasing the mouse button now increases the selection by forms, either forward or backward. It stops when it reaches the start or end of an enclosing form.

Likewise, triple-clicking in the Editor (on GTK+ and Cocoa) and then dragging without releasing the mouse button now increases the selection by lines.

15.10.14 Chrome, Opera and newer versions of Firefox supported

The Chrome and Opera browsers are now supported to display documentation on Linux, FreeBSD, Solaris and AIX. In addition, newer version of Firefox are now supported correctly.

15.11 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. This section is not relevant to LispWorks for Mobile Runtime.

15.11.1 Improved editor handling of byte order mark in Unicode

The Editor now handles the byte order mark (BOM) in UTF-8, UTF-16 and UTF-32 formatted files better. The byte order mark is removed when opening a file in these external-formats and added back when writing a file in UTF-16 and UTF-32 formats. It is not added back when writing in UTF-8 format because that can cause problems for some other programs.

15.11.2 Face objects documented

The system class **editor:face** and the function **editor:make-face** are now documented.

15.12 Foreign Language interface changes

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

15.12.1 Support for vector types

FLI vector types have been added, with names such as **fli:vector-char2**. These correspond to the C / Objective-C vector types that are defined by Clang, which is used on Mac OS X, iOS and FreeBSD and is optionally available on other operating systems.

15.12.2 The fastcall calling convention

The **:calling-convention** keyword argument for **fli:define-foreign-function** and **fli:define-foreign-funcallable** can now have value **:fastcall**, to support the **__fastcall** qualifier in the Visual C and GNU C compilers.

15.12.3 Support for calling variadic functions

fli:define-foreign-function and **fli:define-foreign-funcallable** have a new keyword argument, **:variadic-num-of-fixed**, which needs to be used when calling variadic foreign functions.

15.12.4 Foreign blocks supported on iOS

The foreign block API (see "Block objects in C (foreign blocks)" in the *LispWorks Foreign Language Interface User Guide and Reference Manual*) is now supported on iOS.

15.12.5 Aligning fields in a structure

The **:aligned** option now works in **fli:define-c-struct**. This was documented in LispWorks 7.0 but not implemented.

15.13 COM/Automation changes

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

15.13.1 Support for methods with the vararg attribute

Methods that are declared with the **vararg** attribute in the IDL or type library are now supported in a way that is compatible with the standard COM runtime. You should pass separate arguments in the an Automation call, but expect to receive them as an array in both COM and Automation method implementations.

15.14 Objective-C changes

This section applies only to Macintosh and iOS platforms. See the *LispWorks Objective-C and Cocoa Interface User Guide and Reference Manual* for details.

15.14.1 Support for specifying method argument types for objc:invoke

The types of the method arguments and result can now be specified in a call to **objc:invoke**. This is primarily intended for invoking methods using vector types, which are not compatible with the Objective-C Runtime type encoding API. See "Invoking a method that uses vector types" in the *LispWorks Objective-C and Cocoa Interface User Guide and Reference Manual* for more details.

15.14.2 Accessing Objective-C instance variables

The accessor **objc:objc-object-var-value** now works with variables of all types.

15.15 Common SQL changes

15.15.1 Common SQL support for SQLite

Common SQL now supports SQLite databases.

15.15.2 Using non-ASCII strings on Microsoft SQL Server

Common SQL can now correctly represent all characters when passing a SQL expression containing string literals to Microsoft SQL Server via ODBC. See "Using non-ASCII strings on Microsoft SQL Server" in the *LispWorks User Guide and Reference Manual*.

15.15.3 Added support for fetching Oracle LOBs directly

The **`:binary`** type can be used to fetch Oracle LOBs directly in Common SQL. See "Fetching the contents of the LOBs directly" in the *LispWorks User Guide and Reference Manual*.

15.15.4 varbinary(max)

Fetching data from a column with type **`varbinary(max)`** now works on Microsoft SQL Server.

15.16 KnowledgeWorks changes

This section applies only when you have a license to run KnowledgeWorks. See the *KnowledgeWorks and Prolog User Guide* for details, unless a manual is referenced explicitly.

15.16.1 Profiling KnowledgeWorks forward rules

`hcl:set-up-profiler` now has a *kw-contexts* argument for profiling the rules in KnowledgeWorks contexts. The Profiler tool in the LispWorks IDE also has an option to do this.

15.16.2 Improved performance of forward chaining in KnowledgeWorks

The performance of forward chaining rules in KnowledgeWorks has been improved in some cases.

15.17 Application delivery changes

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

15.17.1 New default for the :keep-modules deliver keyword argument

The **:keep-modules** keyword argument to **lw:deliver** now defaults to **nil**, rather than **(< *delivery-level* 1)**. We recommend using **require** to load all modules before delivery.

15.17.2 Saving a split dynamic library on non-Windows platforms

The **:split** keyword argument to **hcl:save-image** and **lw:deliver** now works for a Lisp dynamic library on all platforms (previously it only worked on Microsoft Windows). Using **:split :resources** for a dynamic library on Mac OS X create a Resources directory adjacent to the dynamic library, as in a framework bundle.

15.17.3 Passing extra linker arguments when making a dynamic library

The functions **hcl:save-image** and **lw:deliver** now take a keyword argument **:dll-extra-link-options**, which allows extra arguments to be passed to the linker when making a dynamic library. This option has existed since LispWorks 6.0 but has not been documented until now.

15.17.4 LispWorks dlls on Linux now require specific versions of the C library symbols

During delivery of a dynamic library on Linux, LispWorks now links to specific versions of symbols in the C library. This makes it more likely that the library will work on older versions of Linux. Libraries created with previous

versions of LispWorks would depend on the version of Linux used to deliver them.

15.17.5 Simplified use of LispWorks as a dynamic library in Java

Delivering LispWorks as a dynamic library to be loaded by Java is made simpler by the new function **lw-ji:setup-deliver-dynamic-library-for-java**. Normally all you need to do is add a call **lw-ji:setup-deliver-dynamic-library-for-java** without arguments before calling **deliver**.

There is an example of this in:

```
(example-edit-file "java/lisp-as-dll/deliv-script")
```

The new function **lw-ji:get-host-java-virtual-machine** returns the host Java virtual machine when it is called in a dynamic library that was delivered with a call **setup-deliver-dynamic-library-for-java**, and the dynamic library was loaded by Java.

15.17.6 Macintosh computers with non-ASCII names

Applications using temporary files will now run on Macintosh computers with non-ASCII names.

15.18 CLIM changes

This section is not relevant to LispWorks for Mobile Runtime.

15.18.1 Evaluation within the process of a sheet

The new function **clim:apply-in-sheet-process** calls a function in the process that is displaying a specified sheet.

15.18.2 API for Drawing with Graphics Ports

The API for drawing with Graphics Ports is deprecated. The functions **clim:draw-gp-image-to-sheet** and **clim:draw-gp-pixmap-to-sheet** have been removed.

15.18.3 Making device font text styles

The function **clim:make-device-font-text-style** has been improved.

15.19 Other changes

15.19.1 ensure-process-cleanup in foreign threads

If the function **mp:ensure-process-cleanup** is called on a foreign thread, the cleanups are now executed after the outermost foreign-callable returns and before return to the foreign code that called it (that is when no Lisp frames remain on the stack).

In LispWorks 7.0 and earlier versions, such a cleanup would never be executed.

15.19.2 Configuration files now explicitly qualify LW and HCL symbols.

The files **config/configure.lisp**, **config/siteinit.lisp**, **private-patches/load.lisp** and **config/a-dot-lispworks.lisp** now explicitly qualify LW and HCL symbols to allows these files to load even if you change the package use list of CL-USER or copy the forms into other files.

15.19.3 Improved performance of sequence functions with :from-end

The Common Lisp sequence functions are now faster when used with a long list and the **:from-end** keyword argument is true.

15.19.4 defparser combined rules

The **parsergen:defparser** macro now allows "combined" rules as a way to group multiple clauses for the same non-terminal.

15.19.5 async-io-state-read-with-checking now resets the old length to 0

The function **comm:async-io-state-read-with-checking** now resets the old length to 0, so the function **comm:async-io-state-old-length** returns 0 in the first invocation of callback.

15.19.6 Profiling new processes

The profiler can now be configured to profile processes that start after the call to **hcl:start-profiling** by using the **:new** value for the **:processes** keyword argument.

15.19.7 The profiler now defaults to monitoring all packages

The function **hcl:set-up-profiler** now defaults to monitoring all packages if the **:packages** and **:symbols** keywords are not supplied.

In addition, if the profiler is invoked before any call to **hcl:set-up-profiler**, it calls **hcl:set-up-profiler** implicitly without any arguments, so will monitor all packages.

15.19.8 Deprecated profiler symbols

The variable **hcl:*profile-symbol-list*** is deprecated.

The functions **hcl:add-symbol-profiler** and **hcl:remove-symbol-profiler** are deprecated. Use **hcl:set-up-profiler** instead.

15.19.9 Naming of flet subfunctions

flet subfunctions now have names like **(flet *name*)** rather than *name*. This is consistent with **labels** subfunctions.

15.19.10 Saving a split dynamic library on non-Windows platforms

The **:split** keyword argument to **hcl:save-image** and **lw:deliver** now works for a Lisp dynamic library on all platforms (previously it only worked on Microsoft Windows). Using **:split :resources** for a dynamic library on Mac OS X create a Resources directory adjacent to the dynamic library, as in a framework bundle.

15.19.11 Passing extra linker arguments when making a dynamic library

The functions **hcl:save-image** and **lw:deliver** now have a keyword argument **:dll-extra-link-options**, which allows extra arguments to be

passed to the linker when making a dynamic library. This option has existed since LispWorks 6.0 but has not been documented until now.

15.19.12 find-regexp-in-string with :case-sensitive :default

The **:case-sensitive** keyword argument to **lw:find-regexp-in-string** now defaults to **nil** and is treated purely as a generalized boolean. In previous releases, the value **:default** was documented as depending on the Editor variable **DEFAULT-SEARCH-KIND**, but this was incorrect.

15.19.13 Reduced memory allocation on Unix platforms with large RLIMIT_NOFILE

LispWorks now uses less memory when waiting for I/O on Unix platforms when the **RLIMIT_NOFILE** resource limit is large (for example 65536 on Linux Debian 8.3).

15.19.14 lw-ji:get-java-virtual-machine can return an existing JVM

The function **lw-ji:get-java-virtual-machine** now returns an existing Java virtual machine if LispWorks already knows about it.

15.19.15 The default JVM library on Windows

On Windows, calling the function **lw-ji:init-java-interface** with **:jvm-library-path t** now checks the registry for the location of the JVM library, using the keys that Oracle document in *Java 2 Runtime Environment for Microsoft Windows*.

15.19.16 cl:software-version now detects Windows 10

The string returned by **cl:software-version** now includes the text "Windows 10" when running on Microsoft Windows 10. Prior to LispWorks 7.1, the description begins with "Windows 8" when running on Windows 10.

15.19.17 Running 32-bit ARM Linux LispWorks on 64-bit ARM Linux

LispWorks (32-bit) for ARM Linux will now run on multiarch 64-bit ARM (AArch64) installations.

15.19.18 LispWorks for Linux on some Intel CPUs

A segmentation violation in **`libpthread.so`** has been fixed when running LispWorks for Linux on certain newer Intel CPUs (those that support the TSX extensions such as Broadwell-EP) and running newer versions of Linux that use them (such as Debian Jessie).

15.19.19 Hibernation on Macintosh computers

Problems have been fixed with timers and event loop errors when hibernating and waking Macintosh computers while running LispWorks.

15.19.20 dbg:log-bug-form message-stream has a different default

The **`:message-stream`** keyword argument to the function **`dbg:log-bug-form`** now defaults to the value of **`cl:*error-output*`**. In LispWorks 7.0 and earlier, it defaulted to the value of **`cl:*debug-io*`**, but this was not documented.

15.19.21 Various Code Coverage bug fixes

Various bugs have been fixed in Code Coverage interface.

15.19.22 Changes in *features*

`:lispworks7.1` is present, **`:lispworks7.0`** 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*.

15.19.23 ASDF version

The supplied ASDF is now version 3.3.0.

15.19.24 Loading old data files

Binary files created with **hcl:dump-forms-to-file** or **hcl:with-output-to-fasl-file** in LispWorks 7.0, LispWorks 6.1, LispWorks 6.0, LispWorks 5.x, LispWorks 4.4 or LispWorks 4.3 can be loaded into LispWorks 7.1 using **sys:load-data-file**.

15.20 Changes in the installers

15.20.1 Package format on Mac OS X

LispWorks 7.1 is now supplied as a **.pkg** file, to be installed using **Installer.app**.

15.20.2 Package format on FreeBSD

LispWorks 7.1 supports FreeBSD 10.x and later and is supplied as a standard package file, in pkg(8) format. Older versions of FreeBSD are not supported.

15.21 Documentation changes

15.21.1 Documentation improved

15.21.2 New self-contained examples

These examples are entirely new:

```
(example-edit-file "capi/applications/price-charting-gt")
(example-edit-file "capi/graphics/graph-color-edges")
(example-edit-file "capi/output-panes/coordinate-origin-fixed")
(example-edit-file "capi/output-panes/fixed-origin-scrolling")
(example-edit-file "java/lisp-as-dll/README.txt")
(example-edit-file "java/lisp-as-dll/LispWorksCaller.java")
(example-edit-file "java/lisp-as-dll/deliv-script")
(example-edit-file "kw/hanoi/hanoi-rules-in-common-prolog")
(example-edit-file "ssl/async-io-client")
(example-edit-file "async-io/udp")
```

Also, the size of the dh_param file used by the ssl examples has been increased from 512 to 1024 bits to support newer servers.

15.21.3 Removed self-contained examples

The pane-can-scroll example has been replaced by:

```
(example-edit-file "capi/output-panes/coordinate-origin-fixed")
```

15.21.4 Corrections

The **:startup-bitmap-file** keyword argument to **lw:deliver** is now correctly documented as defaulting to **:internal**. This changed in LispWorks 7.0 but was not documented. See the *LispWorks Delivery User Guide* for more details.

15.22 Known Problems

15.22.1 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.

15.22.2 Problems with LispWorks for Macintosh

The Motif GUI does not work "out of the box" with Fink because LispWorks does not look for **libXm** etc in **/sw/lib/**.

15.22.3 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 does not 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 Inspector is still displayed.

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.

15.22.4 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* and *fill-style*.

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 does not 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.

Some pinboard code uses **:operation boole-xor** which is not implemented.

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** coordinates of the dialog (for drop-down sheets the coordinates are not relevant, and for dialogs which are separate windows Cocoa forces the window to be in the top-center of the screen).

15.23 Binary Incompatibility

If you have binaries (fasl files) which were compiled using LispWorks 7.0 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 7.1.

Index

D

E

F

G

H

I

J

L

M

N

O

P

R

V

W