LispWorks®

Release Notes and Installation Guide

Version 7.0

Copyright and Trademarks

LispWorks Release Notes and Installation Guide

Version 7.0

March 2015

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

LispWorks Editions 1
LispWorks for Mobile Runtime 3
Further details 4
About this Guide 4

2 Installation on Mac OS X 7

Choosing the Graphical User Interface 7
Documentation 8
Software and hardware requirements 8
Installing LispWorks for Macintosh 9
Starting LispWorks for Macintosh 13
Uninstalling LispWorks for Macintosh 15
Upgrading the LispWorks Edition 15
Upgrading to 64-bit LispWorks 16

3 Installation on Windows 17

Documentation 17
Installing LispWorks for Windows 18
Uninstalling LispWorks for Windows 20

Upgrading the LispWorks Edition 20
Upgrading to 64-bit LispWorks 21

4 Installation on Linux 23

Software and hardware requirements 23
License agreement 25
Software delivery and installer formats 25
Installing LispWorks for Linux 26
LispWorks looks for a license key 32
Running LispWorks 32
Configuring the image 33
Printable LispWorks documentation 33
Uninstalling LispWorks for Linux 33
Upgrading the LispWorks Edition 34
Upgrading to 64-bit LispWorks 34

5 Installation on x86/x64 Solaris 35

Software and hardware requirements 35
Software delivery and installer format 37
Installing LispWorks for x86/x64 Solaris 38
LispWorks looks for a license key 39
Running LispWorks 40
Configuring the image 41
Printable LispWorks documentation 41
Uninstalling LispWorks for x86/x64 Solaris 41
Upgrading the LispWorks Edition 41
Upgrading to 64-bit LispWorks 42

6 Installation on FreeBSD 43

Software and hardware requirements 43
License agreement 45
Software delivery and installer format 45
Installing LispWorks for FreeBSD 46
LispWorks looks for a license key 48
Running LispWorks 48

Configuring the image 49
Printable LispWorks documentation 50
Uninstalling LispWorks for FreeBSD 50
Upgrading the LispWorks Edition 50
Upgrading to 64-bit LispWorks 50

7 Installation on AIX 53

Software and hardware requirements 53
License agreement 55
Software delivery and installer format 55
Installing LispWorks for AIX 55
LispWorks looks for a license key 57
Running LispWorks 57
Configuring the image 58
Printable LispWorks documentation 58
Uninstalling LispWorks for AIX 59
Upgrading the LispWorks Edition 59
Upgrading to 64-bit LispWorks 59

8 Installation on SPARC Solaris 61

Introduction 61
Extracting software from the CD-ROM 61
Moving the LispWorks image and library 63
Obtaining and Installing your license keys 64
Configuring the LispWorks image 65
Using the Documentation 67
Using Delivery, LispWorks ORB, CLIM 2.0, KnowledgeWorks and Common SQL 68

9 Installation of LispWorks for Mobile Runtime 69

Installing LispWorks for Android Runtime 69
Installing LispWorks for iOS Runtime 69

10 Configuration on Mac OS X 71

Introduction 71

License keys 72
Configuring your LispWorks installation 72
Saving and testing the configured image 74
Initializing LispWorks 77
Loading CLIM 2.0 78
The Common SQL interface 78
Common Prolog and KnowledgeWorks 80

11 Configuration on Windows 81

Introduction 81
License keys 82
Configuring your LispWorks installation 82
Saving and testing the configured image 83
Initializing LispWorks 86
Loading CLIM 2.0 86
The Common SQL interface 87
Common Prolog and KnowledgeWorks 88
Runtime library requirement on Windows 88

12 Configuration on Linux, x86/x64 Solaris, FreeBSD & AIX 89

Introduction 89
License keys 90
Configuring your LispWorks installation 90
Saving and testing the configured image 92
Initializing LispWorks 94
Loading CLIM 2.0 94
The Common SQL interface 95
Common Prolog and KnowledgeWorks 96
Documentation on x86/x64 Solaris, FreeBSD and AIX 96

13 Configuration on SPARC Solaris 97

Disk requirements 97
Software Requirements 97
The CD-ROM 98
Installing LispWorks 99

Components of the LispWorks distribution 103
Printing copies of the LispWorks documentation 104
Configuring your LispWorks installation 104
LispWorks initialization arguments 108

14 Troubleshooting, Patches and Reporting Bugs 111

Troubleshooting 111
Troubleshooting on Windows 114
Troubleshooting on Mac OS X 114
Troubleshooting on Linux 115
Troubleshooting on x86/x64 Solaris 117
Troubleshooting on FreeBSD 117
Troubleshooting on SPARC Solaris 118
Troubleshooting on X11/Motif 119
Updating with patches 121
Reporting bugs 124
Transferring LispWorks to a different machine 129

15 Release Notes 131

Keeping your old LispWorks installation 131
Updating your code for LispWorks 7.0 131
Platform support 132
Multiprocessing 134
GTK+ window system 135
New CAPI features 135
New graphics ports features 140
Other CAPI and Graphics Ports changes 141
More new features 146
IDE changes 154
Editor changes 157
Foreign Language interface changes 160
COM/Automation changes 162
Common SQL changes 163
Application delivery changes 164
CLOS/MOP changes 165

CLIM changes 165
Other changes 165
Changes in the installers 168
Documentation changes 168
Known Problems 171
Binary Incompatibility 174

Index 175

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 7.0 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 7.0 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 7.0 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 7.0 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 7.0 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 Further details

For further information about LispWorks products visit

www.lispworks.com

To purchase LispWorks please follow the instructions at:

www.lispworks.com/buy

1.4 About this Guide

This document is an installation guide and release notes for LispWorks 7.0 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 which is distributed separately.

1.4.1 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 7.0. 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.

1.4.2 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.

1.4.3 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 7.0 (32-bit) for Macintosh and LispWorks 7.0 (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 7.0/HTML Documentation.htm`**. If you choose not to install the documentation, you will not be able to access the HTML Documentation from the LispWorks **Help** menu.

The PDF format is suitable for printing. Each manual in the documentation set is presented in a separate PDF file in the LispWorks library under **`manual/offline/pdf`**. The simplest way to locate these PDF files is the alias **`LispWorks 7.0/PDF Documentation`**. To view and print these files, you will need a PDF viewer such as Preview (standard on Mac OS X) or Adobe® Reader® (which can be downloaded from the Adobe website at **`www.adobe.com`**).

2.3 Software and hardware requirements

LispWorks 7.0 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. 240MB 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 Imlib 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. 285MB 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 Imlib if you want to run the deprecated Motif GUI.

2.4 Installing LispWorks for Macintosh

2.4.1 Main installation and patches

The LispWorks 7.0 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.0.x. You need to complete the main installation before adding patches.

2.4.2 Information for Beta testers

Users of LispWorks 7.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.0.

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

2.4.3 Information for users of previous versions

You can install LispWorks 7.0 in the same location as LispWorks 6.1 or previous versions. If you always choose the default install location, a new folder named **`LispWorks 7.0 (32-bit)`** or **`LispWorks 7.0 (64-bit)`** will be created alongside the other versions.

2.4.4 Use an administrator account

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

However, a non-administrator may install LispWorks elsewhere.

2.4.5 Launch the LispWorks installer

Mount the disk image containing the 32-bit LispWorks or 64-bit LispWorks installer. These are called **LispWorks70-32bit.dmg** and **LispWorks70-64bit.dmg** respectively. Simply double-click on the **.dmg** file to mount it.

You will have downloaded these from the **x86-darwin** and **amd64-darwin** folders respectively.

To install LispWorks (32-bit) for Macintosh launch the **LispWorks32bit_Installer** application. To install LispWorks (64-bit) for Macintosh launch the **LispWorks64bit_Installer** application. .

Note: the Personal Edition installer is a **pkg** file. Open it to run the installer in the Mac OS X Installer application.

2.4.6 The Read Me

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

2.4.7 The License Agreement

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

2.4.8 Select Destination

All the files installed with LispWorks are placed in the LispWorks folder, which is named **LispWorks 7.0 (32-bit)**, **LispWorks 7.0 (64-bit)** or **LispWorks Personal 7.0** depending on which edition you are installing. By default, the LispWorks folder is placed in the main **Applications** folder but

you can choose an alternative location during installation by clicking the **Select Folder...** button.

Click **Continue** after selecting a folder.

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

2.4.9 Choose your installation type

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

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

2.4.9.1 The native Mac OS X GUI

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

2.4.9.2 The X11 GTK+ and Motif GUIs

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

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

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

- An X server such as Apple's X11.app, available at `www.apple.com`, and
- one of GTK+ 2 (version 2.4 or higher) or Open Motif 2.3.

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

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

The X11 GUIs are not available for the Personal Edition.

2.4.9.3 The Documentation

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

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

2.4.10 Installing and entering license data

Now click **Install**.

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

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

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

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

2.4.11 Add LispWorks to the Dock

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

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

2.4.12 Finishing up

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

Note: LispWorks needs to be able find its library at runtime and therefore the LispWorks installation should not be moved around piecemeal. If you must move it, move the entire LispWorks installation folder. If you simply want to

run LispWorks from somewhere more convenient, then consider adding an alias.

2.4.13 Installing Patches

After completing the main installation of LispWorks, ensure you install the latest patches which are available for download at **`www.lisp-works.com/downloads/patch-selection.html`**. Patch installation instructions are in the README file accompanying the patch download.

2.4.14 Obtaining X11 GTK+

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

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

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

2.4.15 Obtaining Open Motif and Imlib

LispWorks 7.0 for Macintosh on X11 / Motif requires Open Motif 2.3 and Imlib.

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

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

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

2.5 Starting LispWorks for Macintosh

2.5.1 Start the native Mac OS X LispWorks GUI

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

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

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

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

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

and pass it to the supplied **lispworks-7-0-0-x86-darwin** image.

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, the image is **lispworks-personal-7-0-0-macos-universal**, and **save-image** is not available.

2.5.2 Start the GTK+ LispWorks GUI

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

Follow this session in the X11 terminal:

```
bash-3.2$ cd "/Applications/LispWorks 7.0 (32-bit)"
bash-3.2$ ./lispworks-7-0-0-x86-darwin-gtk
; Loading text file /Applications/LispWorks 7.0 (32-
bit)/Library/lib/7-0-0-0/private-patches/load.lisp
LispWorks(R): The Common Lisp Programming Environment
Copyright (C) 1987-2014 LispWorks Ltd.  All rights reserved.
Version 7.0.0 Beta
Saved by LispWorks as lispworks-7-0-0-x86-darwin-gtk, at 28 Jun
2014 15:05
User dubya on machine.lispworks.com
; Loading text file /Applications/LispWorks 7.0 (32-
bit)/Library/lib/7-0-0-0/config/siteinit.lisp
;  Loading text file /Applications/LispWorks 7.0 (32-
bit)/Library/lib/7-0-0-0/private-patches/load.lisp
; Loading text file /u/ldisk/dubya/.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 installed, you can use LispWorks with the Motif GUI.

You first must load the Motif GUI into the supplied **`lispworks-7-0-0-x86-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 can either:

- Launch the LispWorks installer as described in “Launch the LispWorks installer” on page 10 and select the **Uninstall** option (and then remove any patches), or
- Simply drag the folder named **`LispWorks 7.0 (32-bit)`** or **`LispWorks 7.0 (64-bit)`** to the trash.

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 7.0 (32-bit) for Windows and LispWorks 7.0 (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 7.0 > 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 7.0 > PDF Documentation**. To view and print these files, you will need a PDF viewer such as Adobe® Reader®. If you do not already have this, it can be downloaded from the Adobe website.

3.2 Installing LispWorks for Windows

3.2.1 Main installation and patches

The LispWorks 7.0 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.0.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 7.0 in the same location as LispWorks 6.x, LispWorks 5.x or LispWorks 4.4.5. This is the default installation location.

You can also install LispWorks 7.0 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 7.0 Beta should completely uninstall it before installing LispWorks 7.0. 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 **`LispWorks70-32bit.exe`**. You will have downloaded this from the **`x86-win32`** folder.

To install LispWorks (64-bit) for Windows run **`LispWorks70-64bit.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.

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.

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

When the installation is complete, you can start 32-bit LispWorks on Windows 7 by choosing **Start > All Programs > LispWorks 7.0 (32-bit) > LispWorks 7.0 (32-bit)**, or on Windows 8 by choosing **LispWorks 7.0 (32-bit)** on the Start screen.

Start 64-bit LispWorks similarly.

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

3.3 Uninstalling LispWorks for Windows

To uninstall LispWorks:

1. Select **Programs and Features** in the Control Panel.
2. Select **LispWorks 7.0 (32-bit)** or **LispWorks 7.0 (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 7.0 (32-bit) for x86/x86_64 Linux and LispWorks 7.0 (64-bit) for x86_64 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

Table 4.1 System requirements on Linux

Hardware Requirements	Software Requirements
175MB of disk space for Enterprise Edition (64-bit) plus documentation	GTK+ 2 (version 2.4 or higher) to run the GTK+ GUI. Open Motif 2.2.x and Imlib to run the deprecated Motif GUI
Any modern machine is likely to have sufficient RAM to run LispWorks as distributed.	Firefox or Opera web browser for viewing on-line documentation

Table 4.1 System requirements on Linux

4.1.1 GUI libraries

LispWorks 7.0 for Linux requires that the X11 release 6 (or higher) is installed. It also requires that either GTK+ or Open Motif with Imlib are installed.

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

4.1.1.1 GTK+

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

GTK+ 2 (version 2.4 or higher) is required.

4.1.1.2 Motif

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

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

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

Note: You should be able to run the LispWorks 7.0 Motif GUI and LispWorks 6.0 or LispWorks 5.x simultaneously with Open Motif installed.

4.1.2 Disk requirements

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

The documentation includes printable PDF format manuals. You may delete any of these that you do not need. They are available at **`www.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 7.0 for Linux is supplied as a download. Two formats are provided:

- Red Hat Package Management (RPM) files. 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 7.0 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.0.x. You need to complete the main installation before adding patches.

4.4.2 Information for Beta testers

Users of LispWorks 7.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.0.

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

4.4.3 Installation from the binary RPM file

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

If you already have LispWorks 7.0 Beta installed, please uninstall it before installing this product. See Section 4.9, “Uninstalling LispWorks for Linux”.

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

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

try upgrading to RPM 3.0.2 or greater.

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

4.4.3.1 Installation directories

By default 32-bit LispWorks is installed in **`/usr/lib/LispWorks`** and a symbolic link to the executable is placed in **`/usr/bin/lispworks-7-0-0-x86-linux`**. Similarly, 64-bit LispWorks is installed in **`/usr/lib64/LispWorks`** and a symbolic link to the executable is placed in **`/usr/bin/lispworks-7-0-0-amd64-linux`**. However, the RPM is relocatable, and the **`--prefix`** option can be used to allow the installation of LispWorks in a non-default directory. The default prefix is **`/usr`**.

Note: RPM version 4.2 has a bug which can hinder secondary installations (CLIM, Common SQL, LispWorks ORB or KnowledgeWorks) in a user-specified directory. See “RPM_INSTALL_PREFIX not set” on page 116 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-7.0-n.arch.rpm
```

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

Note: For the Personal Edition, use `lispworks-personal-7.0-*.i386.rpm` wherever `lispworks-7.0-*.i386.rpm` is mentioned in this document. See Section 1.1.1, "Personal Edition" for more information specific to the Personal Edition.

4.4.3.3 Installing or upgrading LispWorks for Linux

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

1. Follow the instructions under Section 4.2, "License agreement".
2. Locate the RPM installation file `lispworks-7.0-n.i386.rpm`.
3. Install or upgrade LispWorks in the standard RPM way, for example:

   ```
   rpm --install lispworks-7.0-n.i386.rpm
   ```

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

   ```
   rpm --install --prefix <directory> lispworks-7.0-n.i386.rpm
   ```

   installs LispWorks in <*directory*>.

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

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

See Section 4.6 for instructions on entering your license details.

4.4.3.4 Installing CLIM 2.0

The following module is packaged as a separate RPM file for installation after the main **lispworks** package. It is available in LispWorks Professional and Enterprise Editions only.

File Distribution	Layered Product
lispworks-clim-7.0-*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-7.0-*n*.i386.rpm	CLIM 2.0
lispworks-kw-7.0-*n*.i386.rpm	KnowledgeWorks
lispworks-corba-7.0-*n*.i386.rpm	LispWorks ORB

Table 4.3 File distributions for layered products in the Enterprise Edition

File Distribution	Layered Product
lispworks-sql-7.0-*n*.i386.rpm	Common SQL

Table 4.3 File distributions for layered products in the Enterprise Edition

Install these modules as described in Section 4.4.3.4.

4.4.3.6 Documentation and saving space

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

Documentation is installed by default in the **lib/7-0-0-0/manual** sub-directory of the LispWorks installation directory.

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

```
rpm --install --excludedocs --prefix <directory>
lispworks-7.0-n.i386.rpm
```

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

```
rpm --install --prefix <directory> --replacepkgs
lispworks-7.0-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 32-bit LispWorks are as follows:

Table 4.4 Files for 32-bit LispWorks

`lw70-x86-linux.tar.gz`	32-bit LispWorks image, modules and examples
`lwdoc70-x86-linux.tar.gz`	Documentation in HTML and PDF formats

Note: The gzipped files for 64-bit LispWorks and 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).

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.lisp-works.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-7-0-0-x86-linux	Personal Edition
lispworks-7-0-0-x86-linux	32-bit LispWorks
lispworks-7-0-0-amd64-linux	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.

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):

lispworks-7-0-0-x86-linux --lwlicenseserial *SERIALNUMBER* --lwlicensekey *LICENSEKEY*

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

```
LispWorks license installed successfully.
```

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

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

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

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/7-0-0-0/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-7.0
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named **lispworks-patches7.0**. 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 7.0 (32-bit) for x86/x64 Solaris and LispWorks 7.0 (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 7.0 for x86/x64 Solaris requires that the X11 release 6 (or higher) is installed. It also requires that either GTK+ or Motif with Imlib are installed.

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

5.1.1.1 GTK+

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

GTK+ 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 7.0 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 **LispWorks70-32bit**.

The package name for 64-bit LispWorks is **LispWorks70-64bit**.

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 **LispWorksPersonal70-32bit**.

5.3 Installing LispWorks for x86/x64 Solaris

5.3.1 Main installation and patches

The LispWorks 7.0 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.0.x. You need to complete the main installation before adding patches.

5.3.2 Information for Beta testers

Users of LispWorks 7.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.0.

See "Uninstalling LispWorks for x86/x64 Solaris" on page 41 for instructions.

5.3.3 Installation directories

32-bit LispWorks is installed by default in **`/opt/LispWorks/lib/LispWorks`** and a symbolic link to the executable is placed in **`/opt/LispWorks/bin/lispworks-7-0-0-x86-solaris`**.

64-bit LispWorks is installed by default in **`/opt/LispWorks/lib/amd64/LispWorks`** and a symbolic link to the executable is placed in **`/opt/LispWorks/bin/lispworks-7-0-0-amd64-solaris`**.

LispWorks Personal Edition is installed by default in **`/opt/LispWorks/lib/LispWorksPersonal`** and a symbolic link to the executable is placed in **`/opt/LispWorks/bin/lispworks-personal-7-0-0-x86-solaris`**.

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

5.3.4 Selecting the correct software package file

The 32-bit LispWorks software package file is called **`LispWorks70-32bit`**.

The 64-bit LispWorks software package file is called **`LispWorks70-64bit`**.

The Personal Edition software package file is called **LispWorksPersonal70-32bit**.

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 LispWorks70-32bit all
```

for 32-bit LispWorks, or

```
pkgadd -d LispWorks70-64bit 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-7-0-0-x86-solaris	Personal Edition
lispworks-7-0-0-x86-solaris	32-bit LispWorks
lispworks-7-0-0-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-7-0-0-x86-solaris --lwlicenseserial SERIALNUMBER
--lwlicensekey LICENSEKEY
```

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

```
LispWorks license installed successfully.
```

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

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

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

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/7-0-0-0/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 7.0 have been installed then you will need to uninstall the patch package, by

   ```
   pkgrm -n LispWorksPatches70-32bit
   ```

   or

   ```
   pkgrm -n LispWorksPatches70-64bit
   ```

2. Then uninstall the main software package containing LispWorks 7.0 by executing:

   ```
   pkgrm -n LispWorks70-32bit
   ```

   or

   ```
   pkgrm -n LispWorks70-64bit
   ```

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 7.0 (32-bit) for FreeBSD and LispWorks 7.0 (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 7.x, or later with compat7x (if you want to run 32-bit LispWorks on FreeBSD 6.x, then please contact Lisp Support)

Table 6.1 System requirements on FreeBSD

Hardware Requirements	Software Requirements
180MB of disk space for 64-bit LispWorks plus documentation	GTK+ 2 (version 2.4 or higher) to run the GTK+ GUI. Open Motif 2.2.x and Imlib to run the deprecated Motif GUI
Any modern machine is likely to have sufficient RAM to run LispWorks as distributed.	Firefox or Opera web browser for viewing on-line documentation

Table 6.1 System requirements on FreeBSD

6.1.1 GUI libraries

LispWorks 7.0 for FreeBSD requires that the X11 release 6 (or higher) is installed.

LispWorks 7.0 also requires that either GTK+ or Open Motif with Imlib are installed.

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

6.1.1.1 GTK+

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

GTK+ 2 (version 2.4 or higher) is required.

6.1.1.2 Motif

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

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

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

6.1.2 Disk requirements

32-bit LispWorks requires about 160MB to install, and 64-bit LispWorks needs 180MB. This includes 110MB of documentation.

The documentation includes printable PDF format manuals. You may delete any of these that you do not need. They are available at **`www.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 7.0 for FreeBSD is supplied as a standard package file 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 **lispworks-32bit-7.0**.

The package name for 64-bit LispWorks is **lispworks-64bit-7.0**.

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 **lispworks-personal-7.0**.

6.4 Installing LispWorks for FreeBSD

6.4.1 Main installation and patches

The LispWorks 7.0 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.0.x. You need to complete the main installation before adding patches.

6.4.2 Information for Beta testers

Users of LispWorks 7.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.0.

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

6.4.3 Installation directories

By default LispWorks is installed in **/usr/local/lib/LispWorks**. A symbolic link to the 32-bit executable is placed in **/usr/local/bin/lispworks-7-0-0-x86-freebsd**. A symbolic link to the 64-bit executable is placed in **/usr/bin/lispworks-7-0-0-amd64-freebsd**. However, the software package is relocatable, and the **-p** option can be used to allow the installation of LispWorks in a user-specified directory. The default prefix is **/usr/local**.

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

6.4.4 Selecting the correct software package file

The 32-bit LispWorks software package file is called

```
lispworks-32bit-7.0.tgz
```

The 64-bit LispWorks software package file is called

```
lispworks-64bit-7.0.tgz
```

The Personal Edition software package file is called

```
lispworks-personal-32bit-7.0.tgz
```

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 lispworks-32bit-7.0.tgz
   ```

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

   ```
   pkg_add -p <directory> lispworks-32bit-7.0.tgz
   ```

   installs LispWorks in *<directory>*.

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

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

See Section 6.6 for instructions on entering your license details.

6.4.6 Installation by non-root users

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

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

```
pkg_add -p installation-directory -R lispworks-32bit-7.0.tgz
```

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

6.4.7 Installing Patches

After completing the main installation of 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 direc-

tory on startup. There is also a symbolic link from the **/usr/local/bin** directory.

The LispWorks executable is named as shown here:.

lispworks-personal-7-0-0-x86-freebsd	Personal Edition
lispworks-7-0-0-x86-freebsd	32-bit LispWorks
lispworks-7-0-0-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-7-0-0-x86-freebsd --lwlicenseserial *SERIALNUMBER* --lwlicensekey *LICENSEKEY*

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

```
LispWorks license installed successfully.
```

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

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

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

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/7-0-0-0/manual/offline** directory contains PDF format versions of the manuals.

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

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

6.9 Uninstalling LispWorks for FreeBSD

To uninstall LispWorks, perform the following steps as root:

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

   ```
   pkg_delete lispworks-patches-32bit-7.0
   ```

   or

   ```
   pkg_delete lispworks-patches-64bit-7.0
   ```

2. Then uninstall the main software package containing LispWorks 7.0:

   ```
   pkg_delete lispworks-32bit-7.0
   ```

   or

   ```
   pkg_delete lispworks-64bit-7.0
   ```

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 7.0 (32-bit) for AIX and LispWorks 7.0 (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 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 7.1 System requirements on AIX

7.1.1 GUI libraries

LispWorks 7.0 for AIX requires that the X11 release 6 (or higher) is installed.

LispWorks 7.0 also requires that either GTK+ or Open Motif with Imlib are installed.

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

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 Imlib (not Imlib2). Imlib version 1.9.13 or later is recommended.

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 7.0 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 7.0 installer contains each of the Editions. Additionally, there may be a patch installer which upgrades LispWorks to version 7.0.x. You need to complete the main installation before adding patches.

7.4.2 Information for Beta testers

Users of LispWorks 7.0 Beta should completely uninstall it (including any patches added to the beta installation) before installing LispWorks 7.0.

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

```
lw70-rs6k-aix.tar.gz
```

The 64-bit LispWorks archive is called

```
lw70-rs6k64-aix.tar.gz
```

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

```
lwdoc70-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`**.

   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 runtime and therefore the LispWorks installation should not be moved around piecemeal. If you simply want to run LispWorks from somewhere more convenient, then consider adding a symbolic link.

See Section 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-7-0-0-rs6k-aix	32-bit LispWorks
lispworks-7-0-0-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-7-0-0-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.

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/7-0-0-0/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-7-0-0-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-7-0-0-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 7.0 for SPARC Solaris and 64-bit LispWorks 7.0 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 7.0 for SPARC Solaris is supplied on a CD-ROM.

32-bit LispWorks 7.0 for SPARC Solaris is supplied in a tar archive also containing the associated products CLIM 2.0, KnowledgeWorks, and LispWorks ORB.

64-bit LispWorks 7.0 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 7.0.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 **lw70-sparc.tar** and **lwdoc70-unix.tar**.

For Sun Sparc (64-bit) you need the files named **lw70-sparc64.tar** and **lwdoc70-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-7-0-0-sparc-solaris is the 32-bit LispWorks image and **lispworks-7-0-0-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/7-0-0-0** alongside the image.

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

1. Put the LispWorks distribution in **/usr/lib/lispworks**. You will then have the LispWorks image at top-level in the **/usr/lib/lispworks** directory, and subdirectories **/usr/lib/lispworks/lib/7-0-0-0**.

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

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

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

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

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

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

   See Section 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/7-0-0-0/config/keyfile.`*hostname***, where ***hostname*** is the name of the host machine on which LispWorks is to run. The **`lib`** directory is

expected by default to be located at **/usr/lib/lispworks/lib** (see Section 8.3 above)

- **lib/7-0-0-0/config/keyfile**, where the **lib** directory is as above.

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

For more details, see "How to obtain keys" on page 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-7-0-0-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-7-0-0-sparc-solaris -build /tmp/config.lisp
   ```

If the image will not run at this stage, it is probably not finding a valid key. See “Obtaining and Installing your license keys” on page 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/7-0-0-0/manual/online** subdirectory of the LispWorks library, and can be accessed via the **Help** menu in the LispWorks IDE.

The PDF format manuals are installed in the `lib/7-0-0-0/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 7.0 for Android Runtime and LispWorks 7.0 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/7-0-0-0/config** subdirectory of the LispWorks installation directory

When the file **lwlicense** is found, it must contain a valid key for the current machine. If you try to run LispWorks without a valid key, a message will be printed to the console reporting that no valid key was found, 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 runtime folder structure, and so on. You can override these settings in your saved image or in your initialization file. You should read through **configure.lisp** .

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

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

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 10.4, below, and Section 10.5, “Initializing LispWorks” for further details.

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

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

Both **configure.lisp** and **a-dot-lispworks.lisp** are preloaded into the image before it is shipped, so if you are happy with the settings in these files, you need not change them. See the example in Section 10.4, below, and Section 10.5, “Initializing LispWorks” for further details.

10.4 Saving and testing the configured image

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

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

10.4.1 Create a configuration file

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made 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 7.0 (32-bit)/LispWorks (32-
   bit).app/Contents/MacOS"
   ```

   or for the X11 / GTK+ LispWorks image:

   ```
   % cd "/Applications/LispWorks 7.0 (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-7-0-0-x86-darwin -build /tmp/save-config.lisp
   ```

   or

   ```
   % ./lispworks-7-0-0-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-7-0-0-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 7.0 (32-bit)/LispWorks (32-
bit).app/Contents/MacOS/lispworks-7-0-0-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 7.0 (32-bit)/LispWorks (32-
bit).app/Contents/MacOS/lispworks-7-0-0-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 runtime, for example:

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

or

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

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

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 runtime folder structure, and so on. You can override these settings in your saved image or in your initialization file. You should read through **configure.lisp** .

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

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

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 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-7-0-0-x86-win32.exe -
   build C:\temp\save-config.lisp
   ```

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

Saving the image takes some time.

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

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

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-7-0-0-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-7-0-0-x86-win32.exe -init
my-lisp-init
```

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

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

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

```
C:\Program Files\LispWorks>lispworks-7-0-0-x86-win32.exe -init -
-siteinit -
```

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

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

11.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 7.0 with

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

and the CLIM demos with

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

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

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

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

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 runtime call:

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

and then you can connect to any installed ODBC datasource.

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

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

and at runtime call:

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

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

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/7-0-0-0/config** subdirectory of the LispWorks installation directory

When the file **lwlicense** is found, it must contain a valid key for the current machine. If you try to run LispWorks without a valid key, a message will be printed reporting that no valid key was found, 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 runtime folder structure, and so on. You can override these settings in your saved image or in your initialization file. You should read through **configure.lisp** .

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

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

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 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-7-0-0-x86-linux -build /tmp/save-config.lisp
   ```

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

Saving the image takes some time.

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

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

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-7-0-0-x86-linux -init my-lisp-init
```

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

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

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

```
% lispworks-7-0-0-x86-linux -init - -siteinit -
```

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

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

12.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 7.0 with

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

and the CLIM demos with

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

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

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

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

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 runtime call:

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

and then you can connect to any installed ODBC datasource.

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

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

and at runtime call:

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

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

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/7-0-0-0/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/7-0-0-0/manual/online** directory of the LispWorks library.

13.2 Software Requirements

The LispWorks 7.0 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 7.0 CD-ROM which contains the LispWorks products you have bought, and how to mount it.

13.3.1 The LispWorks 7.0 CD-ROM

The CD-ROM contains images for LispWorks 7.0 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_70/
```

To unmount:

```
umount /cdrom/lw_70/
```

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 7.0.

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_70/**, you would type

```
tar -xof /cdrom/lw_70/lw70-sparc.tar
```

13.4.1.3 SPARC Solaris (LispWorks 32-bit)

The files you need to unpack for LispWorks (32-bit) on Solaris are **lw70-sparc.tar** and **lwdoc70-sparc.tar**.

The LispWorks image is:

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

13.4.1.4 SPARC Solaris (LispWorks 64-bit)

The files you need to unpack for LispWorks (64-bit) on Solaris are **lw70-sparc64.tar** and **lwdoc70-sparc64.tar**.

The LispWorks image is:

```
./lispworks-7-0-0-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/7-0-0-0/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/7-0-0-0/config**.

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

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-7-0-0-sparc-solaris** is the 32-bit LispWorks 7.0 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 runtime directory structure (basically, everything except the image file) should be somewhere publicly readable: **/usr/lib/lispworks**, by default. If there is not enough room in any of the normal publicly accessible locations, you could put a symbolic link there pointing to the installation directory in a partition with more disk space. The installation directory must contain a subdirectory called **lib/7-0-0-0/**.

Among the directories on this subdirectory are the following:

- **config** — various files that can be adjusted in order to customize the image (see Section 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/7-0-0-0/**, although you may place the **lib** directory somewhere else.

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

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

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

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

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

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

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

On startup, the image loads **siteinit.lisp** and your **.lispworks** file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 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-7-0-0-sparc-solaris -init - -siteinit -
   ```

   If the image will not run at this stage, it is probably not finding a valid key. See “Keyfiles and how to obtain them” on page 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/7-0-0-0/load-on-demand`** under **`*lispworks-directory*`**.

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

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

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

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 runtime initialization, or something else that assumes the application is already running.

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

Such initializations should be done at the start of the runtime phase, as described in "Separate runtime initializations from the build phase" on page 13.

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 & Windows 8

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\dubya\AppData\Local\VirtualStore\Program Files
(x86)\LispWorks\lib\7-0-0-0\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 Default installation requires administrator on Mac OS X

To install LispWorks in the default installation location under **/Applications** you must log on as an administrator. So it is usually best to run the **LispWorks32bit_Installer** or **LispWorks64bit_Installer** application as an administrator - the account you created when setting up your Macintosh is an administrator, for instance.

However, a non-administrator may install LispWorks elsewhere.

14.3.2 Text displayed incorrectly in the editor on Mac OS X

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

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

Select the font for Editor and Listener tools by **LispWorks > Preferences... > Environment > Styles > Editor Font**.

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 119 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 7.0 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-7.0-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-7.0-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-7.0-1
#
```

This is only a problem when LispWorks itself was installed in a non-default location (that is, using the **`--prefix`** RPM option). You would then want to supply that same **`--prefix`** value when installing the secondary rpm. A bug in RPM means that a required environment variable **`RPM_INSTALL_PREFIX`** is not set 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-7.0-1.i386.rpm
```

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

```
setenv RPM_INSTALL_PREFIX /usr/lisp
rpm --install --prefix /usr/lisp lispworks-clim-7.0-1.i386.rpm
```

14.4.3 Using multiple versions of Motif on Linux

The version of Open Motif required by LispWorks 7.0 with the Motif GUI may not be compatible with other applications (including LispWorks 4.2). It is however compatible with LispWorks 6.1, LispWorks 5.x, LispWorks 4.4 and 4.3, so you for example you should be able to run LispWorks 7.0 and LispWorks 6.1 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 file **libXm.so** then set **LD_LIBRARY_PATH** to include <*motiflibdir*>.

Note: to find out which version of libXm your LispWorks 7.0 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 171 and "Troubleshooting on X11/Motif" on page 119.

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 119 below for issues specific to X11/Motif.

14.6.1 Poor latency when using multiple threads

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

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 119 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:

```
lwdoc70-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/7-0-0**) 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 **~/Lisp-works** 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 7.0/lispworks-7-0-0-macos-
universal-motif can't open library:
/usr/local/lib/libXm.4.dylib (No such file or directory, errno
= 2)
.
```

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

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\0001.64ofasl (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.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/7-0-0-0/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 7.0/Library/lib/7-0-0-0/private-patches/my-patch.xfasl

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

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

On Linux:

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

On SPARC:

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

as in the commented example there. **load-all-patches** loads this file, and hence all the private patches listed therein.

You may choose to save a reconfigured image with the new patch loaded - for details see the instructions in Section 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. 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 7.0 (32-bit)/LispWorks (32-
   bit).app/Contents/MacOS/lispworks-7-0-0-x86-darwin" -init
   init.lisp > lw.out
   ```

   where `%` denotes a Unix shell prompt.

   On Windows:

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

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

   On Linux:

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

   where `%` denotes a Unix shell prompt.

   On UNIX (SPARC in this example):

   ```
   % /usr/lib/lispworks/lib/7-0-0-0/config/lispworks-7-0-0-sparc-
   solaris -init init.lisp > lw.out
   ```

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

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/7-0-0-0/private-patches` directory) for this version of LispWorks, move them and your `private-patches/load.lisp` file to the corresponding location in the new installation.

4. Test the new installation by running LispWorks and check the patch banner in the output of **Help > Report Bug**. It should be identical to the original installation. If it 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/7-0-0-0/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 33
- "Uninstalling LispWorks for x86/x64 Solaris" on page 41
- "Uninstalling LispWorks for FreeBSD" on page 50

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 7.0 in the same directory as previous versions such as LispWorks 6.1. This is because most of the 7.0 files are stored in a subdirectory called **lib/7-0-0-0**.

Binaries produced by **cl:compile-file** in previous versions of LispWorks do not load into a LispWorks 7.0 image. You must recompile all your code with the LispWorks 7.0 compiler.

15.2 Updating your code for LispWorks 7.0

Check through these release notes for things you need to update in code that already works in LispWorks 6.1.

If you are updating code that works only in versions earlier than LispWorks 6.1, 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 for Android and iOS

With LispWorks for Mobile Runtime you can now build a "headless" LispWorks runtime for Android or iOS devices. Connect this with a native GUI to create a mobile app with LispWorks inside.

LispWorks for Android Runtime supports all of ANSI Common Lisp.

LispWorks for iOS Runtime supports all of ANSI Common Lisp except for the compiler (because it is not possible to create executable code on-the-fly on iOS).

Your app can include your own Lisp code, LispWorks modules such as KnowledgeWorks, and third party Common Lisp libraries. However note that CAPI is not supported on iOS or Android, therefore any GUI part of your app will need to be written using the native API. For more information see the chapters "iOS interface" and "Android interface" in the *LispWorks User Guide and Reference Manual*.

15.3.2 AIX/PowerPC implementation

LispWorks for AIX / PowerPC is now available. There are 32-bit and 64-bit implementations.

15.3.3 ARM Linux implementation

LispWorks (32-bit) for ARM Linux is now available.

15.3.4 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.

15.3.5 Code signing LispWorks images

15.3.5.1 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.

15.3.5.2 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.

15.3.5.3 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.

15.3.6 Mac OS X universal binaries no longer supported

LispWorks 7.0 for Macintosh does not support the PowerPC architecture or universal binaries.

The LispWorks image supplied is named according to its (single) architecture, for example **`lispworks-7-0-0-x86-darwin`** and **`lispworks-7-0-0-amd64-darwin`**. It cannot build a universal binary runtime.

LispWorks 6.1 and some earlier versions were supplied as a universal binary image and could build universal binary runtimes.

The functions **hcl:save-universal-from-script**, **hcl:create-universal-binary**, **hcl:save-argument-real-p** and **hcl:building-universal-intermediate-p** are all deprecated.

15.3.7 Older platforms

LispWorks 7.0 does not support HP-UX.

LispWorks no longer supports Mac OS X 10.3 or 10.4, and does not support any version of Mac OS X on PowerPC.

LispWorks 7.0 (32-bit) for Macintosh supports Mac OS X 10.5 and later on Intel.

LispWorks 7.0 (64-bit) for Macintosh supports Mac OS X 10.5 and later on Intel.

15.4 Multiprocessing

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

This section describes changes made in the multiprocessing interface since version 6.1.

15.4.1 Modifying hash tables with multiprocessing

New APIs allow you to ensure hash table entries in a thread-safe way. This means attempting to get a value, and if that fails setting and returning a new value (potentially having constructed the new value in a complicated way). See "Modifying a hash table with multiprocessing" in the *LispWorks User Guide and Reference Manual*.

15.4.2 Communication between processes

The functions **mp:current-process-send** and **mp:mailbox-wait** are new. Also the documentation of interrupts and communication between processes has been improved. See sections "Processes basics" and "Communication

between processes and synchronization" in the *LispWorks User Guide and Reference Manual*.

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 171.

15.5.1 Using Motif instead of GTK+

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

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

To use LispWorks 7.0 with Motif you also need Imlib installed, as described earlier in this manual.

15.6 New CAPI features

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

15.6.1 Touch gestures

On Cocoa and Windows the **`capi:output-pane`** *input-model* can now contain specifications for multi-touch gestures that come from devices that can generate them (trackpad or touchscreen). These include zoom, rotate, pan, swipe (Cocoa only), two finger tap (Windows only), press and tap (Windows only), and the beginning or end of a sequence of gestures.

15.6.2 editor-pane allows more fonts on Cocoa

You can now use any available font in **capi:editor-pane** in the Cocoa GUI of LispWorks for Macintosh.

In LispWorks 6.1 and earlier versions, you could select only fonts whose widths are (almost) integral, which is not generally guaranteed.

15.6.3 Full screen windows on Cocoa

CAPI now supports full screen windows on Mac OS X 10.7 and later, automatically hiding the menu bar and dock. Use the value **:can-full-screen** in the *window-styles* list (see the **capi:interface** manual page).

15.6.4 Transient display on output panes

The new Cached Display interface allows you to add transient drawing on top of the permanent drawing of a **capi:output-pane**. Typically this will be used to give visual feedback to the user as she drags the mouse while pressing a button in order to select something. You could draw a rectangle, or something more complex.

The Cached Display interface is useful in cases where the *display-callback* is too slow to add the transient graphics.

For an overview see "Transient display on output-pane and subclasses" in the *CAPI User Guide and Reference Manual*.

15.6.5 Scroll hints for output panes and layouts

You can now specify scroll parameters by supplying hints when creating a **capi:output-pane** or a **capi:layout**.

See the section "Scroll values and initialization keywords" in the *CAPI User Guide and Reference Manual*.

15.6.6 Minima for column width and row height

The new **capi:grid-layout** initargs **:min-column-width** and **:min-row-height** can be specified to provide minimum values for the width of each column and the height of each row.

15.6.7 Improvements in multi-column-list-panel

The number of columns in a **capi:multi-column-list-panel**, their titles and what they show can be changed after the pane is displayed using the new function **capi:modify-multi-column-list-panel-columns**.

Also the *column-function* of a **capi:multi-column-list-panel** can now be a list of function designators, each of which is called with the item to generate the object for the corresponding column.

15.6.8 Accessing a sorted-object to determine the sorting

The new keyword argument **:object-sort-caller** for **capi:make-sorting-description** allows you to access the **capi:sorted-object** to decide how to do the sorting.

15.6.9 Pinboard display-callback changed - no longer use capi::pinboard-pane-display

The new generic function **capi:pinboard-layout-display** is now the default *display-callback* of **capi:pinboard-layout**. Programmers who specialized the internal **capi::pinboard-pane-display** (which is the default *display-callback* in LispWorks 6.1 and earlier versions) must now change to **capi:pinboard-layout-display**.

15.6.10 Modifying pinboard display

The new function **capi:draw-pinboard-layout-objects** draws the pinboard objects within a specified region of a pinboard layout. It is useful when you want to have your own *display-callback* for a **capi:pinboard-layout** or a subclass.

15.6.11 Modifying resize behavior of an output pane

The new generic function **capi:output-pane-resize** is called when a **capi:output-pane** is resized. You may need to define your own method if you define your own subclass of **capi:output-pane** which needs to do something when resizing, and you want to allow different *resize-callback*s for individual instances of your class.

15.6.12 Cleaning up objects associated with a pinboard layout

The functions **capi:record-dependent-object** and **capi:unrecord-dependent-object**, together with the generic function **capi:destroy-dependent-object** form a new mechanism for destroying objects when a **capi:pinboard-layout** is destroyed.

15.6.13 Modifier change callbacks in output-pane

capi:output-pane now supports modifier change callbacks, allowing your application to respond when the state of **Control**, **Shift**, **Meta**, **Command** or **Caps Lock** changes.

For the details, see the description of **capi:output-pane** *input-model*.

15.6.14 Detecting modifier state

The new function **capi:pane-modifiers-state** returns an integer describing which modifier keys (**Control**, **Shift**, **Meta**, **Command** or **Caps Lock**) are currently active.

15.6.15 Querying fonts, text and image size, before display

The new function **capi:create-dummy-graphics-port** allows you to query fonts and measure text or images, without displaying anything on screen.

15.6.16 Finding an active screen

capi:convert-to-screen now accepts the special argument **:if-any**, which finds a screen if there is any active screen.

15.6.17 Predicate for availability of browser-pane

The new function **capi:browser-pane-available-p** tests whether **capi:browser-pane** is implemented for a specified screen.

15.6.18 Restoring disabled display of a pane

The new functions **capi:pane-restore-display** and **capi:pane-can-restore-display-p** can allow you to restore a pane's display which has been disabled, typically after an error in the display callback.

15.6.19 New default menu accelerators on Windows and GTK+

On Microsoft Windows and GTK+, menu items named **File > Close** and **File > Exit** now automatically get accelerator keys as follows:

File > Close	**Ctrl+W**
File > Exit	**Ctrl+Q**

For the full set of defaults see "Standard default accelerators" in *CAPI User Guide and Reference Manual*.

15.6.20 Cancelling a popup menu

The new function **capi:popup-menu-force-popdown** cancels a popup menu if it is currently displayed. It can be called from any process, such as a timer.

15.6.21 New initargs to toolbar and toolbar-component

capi:toolbar now supports new initargs **:names** and **:texts**, which allow you to specify the *name* and *text* for each implicitly-created **capi:toolbar-button**.

capi:toolbar-component also now supports the new initargs **:names** and **:texts**. Note that buttons inside an interface toolbar (created with the **capi:interface** initarg **:toolbar-items**) must have unique names.

15.6.22 Display a transient message in a non-focus window

The new function **capi:display-non-focus-message** displays a message for a short period of time in a window which does not take the input focus, to notify the user of something that does not actually require her attention.

15.6.23 Positioning a non-focus list prompter

The function **capi:prompt-with-list-non-focus** now accepts keyword arguments **:right** and **:bottom** which allow you to supply an integer specifying the positioning of the right or bottom of the window. *right* and *bottom* can both also be the keyword **:center** which means center the window relative to the owner window, in the x or y dimension.

15.6.24 Experiment with your code displayed as a dialog

The development utility function **capi:contain** can now display the interface that it creates as a dialog rather than an ordinary window. See the keyword argument **:as-dialog**.

15.6.25 Filters in multiple file prompter

The **:filters** keyword argument to **capi:prompt-for-files** is now implemented on Cocoa and GTK+.

In LispWorks 6.1 and previous version it works only on Microsoft Windows.

15.6.26 Stop an ongoing composition

The new function **capi:output-pane-stop-composition** stops an ongoing composition, returning the currently composed string. You will typically need to call it when a gesture that is not processed by the input method (for example a mouse click) changes the interaction such that it no longer makes sense to continue the composition.

15.7 New graphics ports features

Unless otherwise stated, for details see the Graphics Ports chapters in the *CAPI User Guide and Reference Manual*.

15.7.1 New Graphic Tools API

The **lw-gt** package contains a basic implementation of an API for drawing resizable objects, currently simple bar charts and graphs.

This new API is not polished or complete, so we would be pleased to read your ideas for improvements and extensions. Documentation is in the chapters "Graphic Tools drawing objects" and "LW-GT Reference Entries" in the *CAPI User Guide and Reference Manual*.

15.7.2 invalidate-rectangle-from-points

The new function **gp:invalidate-rectangle-from-points** invalidates a rectangle specified by two points, causing it to be redisplayed.

15.7.3 port-owner

The new function **gp:port-owner** returns the port owner of a graphics port.

15.8 Other CAPI and Graphics Ports changes

15.8.1 Tree view RSS reader example

A new example illustrates how to combine an XML parser with **capi:tree-view** to display an RSS file:

```
(example-edit-file "capi/applications/rss-reader")
```

15.8.2 Windows 8 style improvements

The class **capi:multi-column-list-pane** with titles and subclasses of **capi:output-pane** and **capi:layout** are now displayed on Windows 8 without a recessed border, to match the default theme. This change is also visible in the LispWorks IDE.

15.8.3 Premultiplication of Image Access pixel values

The Graphics Ports documentation has been changed to clarify that color values that are received and set using the Image Access API are *premultiplied*, which means that the value of each of the three components (Red, Green and Blue) is already multiplied by the Alpha value. This is different from the way colors are represented elsewhere in Graphics Ports.

The new functions **color:color-to-premultiplied** and **color:color-from-premultiplied** can be used to convert between premultiplied colors and ordinary colors, though they lose some precision in the process.

15.8.4 Transparency from an external image

The *transparent-color-index* argument to **gp:read-external-image** can now be a cons:

(***index*** **.** ***new-color***)

where *new-color* is a color specification that is converted to the color to use.

new-color can also be **:transparent**. On Mac OS X, Microsoft Windows and GTK+ this makes it truly transparent.

15.8.5 Flagging an editor-pane for application-only use

The new **capi:editor-pane** initarg **:flag** allows you to ensure that creating your instance of **capi:editor-pane** creates a new **editor:buffer** object, which is killed when your editor pane is destroyed.

Buffers created in this way do not appear in IDE windows such as the Editor tool's Buffers view or the Editor command **Select Buffer**.

The **:flag** keyword argument of **capi:modify-editor-pane-buffer** is deprecated.

15.8.6 Text printing now makes a new page for formfeed

The CAPI text printing functions such as **capi:print-text** and **capi:print-file** now start a new page when a line consisting of just a formfeed character (ASCII 12) is found in the text.

15.8.7 text-input-pane default limit increased on Windows

On Microsoft Windows, a default limit on the number of characters in a **capi:text-input-pane** has been increased. The pane can now contain more text when *max-characters* is **nil**, though you can still specify *max-characters* directly.

15.8.8 draw-image width and height default values

The **gp:draw-image** arguments *from-width* and *from-height* now default to the size of the image. In addition, *to-width* defaults to *from-width* and *to-height* defaults to *from-height*.

In LispWorks 6.1 and earlier releases, the *to-width* and *to-height* arguments defaulted to the size of the image and *from-width* defaulted to *to-width* and *from-height* defaulted to *to-height*.

15.8.9 find-matching-fonts weight and slant default values on Windows

gp:find-matching-fonts now defaults the *weight* and *slant* attributes to value **:wild** on Microsoft Windows. This matches what it has always done on other platforms.

15.8.10 Temporary editor buffer for an editor-pane

capi:editor-pane now accepts *buffer-name* initarg value **:temp**.

This creates a temporary buffer which, except when actually editing files, is what you normally need.

15.8.11 Representing modified keyboard input

Character bits attributes are no longer supported hence you can no longer supply characters with bits in the *input-model* of **capi:output-pane** (and its subclasses such as **capi:editor-pane** and **capi:graph-pane**).

If you need to represent keyboard input with modifier keys, use **sys:gesture-spec** objects.

15.8.12 Changing the input model of a graph pane

Using the initarg **:input-model** in **capi:graph-pane** now overrides the existing *input-model*. If you want to use **capi:graph-pane** with a modified input model, you need to avoid passing **:input-model**, and instead use the **capi:output-pane-input-model** accessor. The best way of doing this is to have your own subclass of **capi:graph-pane** and do it in the **cl:initialize-instance** method of your subclass. Note that to override the

input-model of the **capi:graph-pane** in cases of clashes you need to prepend your mappings, for example:

```
(defclass my-graph-pane (graph-pane) ())

(defmethod initialize-instance :after ((pane my-graph-pane) &key)
  (setf (output-pane-input-model pane)
        (append *my-input-model* (output-pane-input-model pane))))
```

capi:graph-pane now behaves like all other subclasses of **capi:output-pane** in this respect.

15.8.13 Additional space inside frames

Additional space is added inside the frame when using the **capi:titled-object** initarg **:title-position** with value **:frame**.

Using **:title-position :frame** now adds a border around the contents to create some space between it and the frame. This border is only added if the contents have *internal-border* **nil**.

15.8.14 title-pane does not support :pane-menu on Microsoft Windows.

This is not a change of functionality but a clarification of a restriction in LispWorks for Windows. If you need interaction, use **capi:display-pane** or **capi:text-input-pane** with **:pane-menu** and **:enabled :read-only**.

15.8.15 File dialog placement on Windows XP

On Windows XP the file dialog now appears on the same monitor as its owner. Note however that LispWorks 7.0 does not support Windows XP.

15.8.16 Graphics Port classes documented

capi:printer-port is a graphics port, used by **capi:with-print-job** when a pane is not supplied. **capi:printer-port** is not new, but is now documented in its own right.

gp:graphics-port-mixin is now documented.

capi:metafile-port is now exported and documented.

15.8.17 Bitmap ports deprecated

gp:bitmap-port, **gp:create-bitmap-port**, **gp:with-bitmap-port** are deprecated. They are implemented only on Motif, they remain undocumented and are not going to be supported on other platforms.

15.8.18 Dithers deprecated

gp:with-dither, **gp:make-dither**, **gp:initialize-dithers** and **gp:dither-color-spec** are deprecated. Dithers do not affect drawing or anti-aliasing.

15.8.19 Font single- and dual-width predicates deprecated

gp:font-dual-width-p and **gp:font-single-width-p** are deprecated.

15.8.20 collection-search deprecated

capi:collection-search is deprecated. Use **capi:collection-find-string** instead.

15.8.21 pane-close-display deprecated

capi:pane-close-display is deprecated.

15.8.22 pane-adjusted-position, pane-adjusted-offset deprecated

capi:pane-adjusted-position and **capi:pane-adjusted-offset** are deprecated.

15.8.23 pinboard-pane-position and pinboard-pane-size deprecated

capi:pinboard-pane-position and **capi:pinboard-pane-size** are deprecated, but are retained in LispWorks 7.0 for backwards compatibility.

Please use **capi:static-layout-child-position** and **capi:static-layout-child-size** instead. These newly-documented functions do just the same.

15.8.24 pane-string deprecated

capi:pane-string is deprecated. Use the accessor **capi:editor-pane-text** instead.

15.8.25 capi:*update-screen-interfaces-hooks* deprecated

capi:*update-screen-interfaces-hooks* is deprecated as we do not believe it is useful. If you use it, please contact Lisp Support.

15.8.26 interactive-pane synonyms dropped

In LispWorks versions 4.x, 5.x and 6.x the class **capi:interactive-pane** and its accessors have synonyms for backwards compatibility with the LispWorks 3.x names **capi:interactive-stream** and so on.

This synonym and the accessor synonyms **capi:interactive-stream-top-level-function** and **capi:interactive-stream-stream** do not exist in LispWorks 7.0.

capi:interactive-pane and its accessors **capi:interactive-pane-top-level-function** and **capi:interactive-pane-stream** are still supported and their functionality is not affected by this change.

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 Java interface

The LispWorks Java interface allows you to:

- Define "Java Callers" which are Lisp functions that call Java methods or constructors, or access Java fields. You can define specific callers or import an entire Java class.
- Make and access Java arrays.
- Make calls from Java into Lisp, either by calling Lisp directly or making proxies that implement some Java interface.

- Access Java objects.
- Integrate Java in a limited way with CLOS.
- Make socket streams using Java sockets.

See the Chapter "Java interface" in the *LispWorks User Guide and Reference Manual*.

15.9.2 Android interface

The Android interface allows your Android project, written in Java, to link to a dynamic library runtime built with LispWorks for Android Runtime and thus create an Android app with LispWorks inside.

It uses the LispWorks Java interface.

See the Chapter "Android interface" in the *LispWorks User Guide and Reference Manual*.

15.9.3 iOS interface

The iOS interface allows your Xcode project to link to a static object runtime built with LispWorks for Android Runtime and thus create an iOS app with LispWorks inside.

It uses the LispWorks Objective-C interface.

See the Chapter "iOS interface" in the *LispWorks User Guide and Reference Manual*.

15.9.4 Unicode support extended

The character implementation in LispWorks now covers the full range of the Unicode standard, including the supplementary planes in addition to the Basic Multilingual Plane (BMP). This involves the following changes:

`cl:char-code-limit` is now **`#x110000`**, which covers exactly the Unicode range. The surrogate code points (codes **`#xd800`** to **`#xdfff`**) are illegal as character codes.

`cl:code-char` accepts integers from 0 below **`cl:char-code-limit`**. Other values cause an error. For codes in the surrogate range it returns **`nil`**. Reading

characters from streams and converting characters from foreign strings can generate characters in the whole range (depending on the external format used), and can never generate character objects corresponding to surrogate code points.

Character bits and font attributes (as specified in *Common Lisp: the Language (2nd Edition)* are no longer supported. **char-bits**, **char-bit**, **(setf char-bit)**, **char-bits-limit**, **char-font** and **char-font-limit** have been eliminated from the LISPWORKS package. The syntax for reading characters with bits is now illegal and trying to read such a character (for example **#\c-a**) will signal an error.

LispWorks 6.1 and earlier versions also accept some special characters representing keystrokes that do not correspond to Unicode characters, such as **#\F1** and **#\Home**. These are illegal in LispWorks 7.0 and later and will signal an error too.

Use strings rather than characters to define LispWorks Editor key sequences with modifiers. See “Editor bindings for keystrokes with modifiers” on page 158 for examples.

In other situations when you previously used character bits, you should use Gesture Spec objects (see **sys:make-gesture-spec** and **sys:coerce-to-gesture-spec**). The function **sys:gesture-spec-to-character** will now signal an error if the gesture spec has bits or corresponds to a non-character key like **F1**, unless the new *errorp* argument is supplied as **nil**.

lw:text-string and **lw:simple-text-string** now can store the full range of Unicode characters.

lw:simple-char is now a synonym for **cl:character**, and is deprecated.

16-bit characters and 16-bit strings are implemented by the new types **lw:bmp-char** and **lw:bmp-string** and **lw:simple-bmp-string** (BMP is Basic Multilingual Plane, comprising the Unicode code points **0** to **#xffff**).

The implementation of 8-bit characters has not changed.

Performance note: **lw:text-string** and **lw:simple-text-string** consume 32 bits per character now, but in LispWorks 6.1 and earlier versions they consume only 16 bits per character. If you have an application that uses many 16-bit strings and you are worried about memory size, you may consider

using strings of type **lw:bmp-string** instead of **lw:text-string**. That will work provided that all the characters you ever use have codes less than **#x10000**. If the codes are all below **256**, you can use **cl:base-string** instead.

15.9.4.1 External formats :unicode and :bmp

External format **:unicode** (synonym **:utf-16**) now includes the entire Unicode range. The new external format **:bmp** means only the BMP (which is the meaning of **:unicode** in LispWorks 6.1 and earlier versions).

15.9.5 Case-insensitive character comparisons and categories

The functions **lw:unicode-string-equal**, **lw:unicode-string-greaterp**, **lw:unicode-string-not-lessp**, **lw:unicode-char-equal**, **lw:unicode-alpha-char-p** and so on have been updated to follow the rules specified in Unicode 6.3.0.

15.9.6 Code Coverage

Code Coverage allows you to compile your code with code execution counters which record when each piece of code is executed. After exercising your code you use a web browser or the LispWorks IDE tools to visualize which parts of your program were executed and how frequently.

Alternatively you can compile your code with a binary flag to record simply whether each piece was executed or not.

See "Code Coverage" in the *LispWorks User Guide and Reference Manual*, and “New Code Coverage tool” on page 155.

15.9.7 Asynchronous I/O on socket streams

You can now perform socket I/O operations that invoke a callback when they are complete, rather than synchronously calling a function that returns a value (like **cl:read-line**). This allows many operations to run in a single thread. See "The Asynchronous I/O API" in the *LispWorks User Guide and Reference Manual*.

UDP sockets are now supported.

15.9.8 Funcall a function asynchronously

The new functions **mp:funcall-async** and **mp:funcall-async-list** call a function as **cl:funcall** would do, but asynchronously.

15.9.9 Fast access to files in a directory

The new function **hcl:fast-directory-files** is a faster way to access files than **cl:directory**, especially in situations when you need to filter based on simple features such as size and access time, or filter based on the name in a more complex way than **cl:directory** can.

15.9.10 Unlocked queues

The new **hcl:unlocked-queue** is a FIFO queue that is unlocked, not thread-safe and does not have waiting functionality, but is faster than **mp:mailbox**.

hcl:unlocked-queue is useful when accessing the queue is always done together with other operations that need to be "atomic", so that you need a lock around them anyway, or when adding and removing items is done in the same process.

15.9.11 find-regexp-in-string improvement

lw:find-regexp-in-string can now return a vector specifying the matches of **\(** and **\)** in the pattern.

15.9.12 Optimal 64-bit arithmetic

The new INT64 API provides a way to perform optimal raw 64-bit arithmetic. See the section "Fast 64-bit arithmetic" in the *LispWorks User Guide and Reference Manual*

15.9.13 KOI8-R encoding supported

A new external format and charset **:koi8-r** has been added to support the KOI8-R (RFC 1489) encoding. You can use this in file I/O as well as the FLI functions.

15.9.14 run-shell-command implemented for Windows

The function **sys:run-shell-command** is now implemented for Microsoft Windows, as well as Unix / Linux / Mac OS X. It allows you to call executables and DOS or Unix shell commands from Lisp code.

15.9.15 Change to run-shell-command with :save-exit-status t

When **sys:run-shell-command** is called with **:wait nil** and **:save-exit-status t**, it now always returns three values (*stream*, *error-stream*, *process*) even if none of the *input*, *output* or *error-output* arguments have the value **:stream**. This allows the exit status of the process to be recovered by calling **sys:pipe-exit-status** on *stream*.

15.9.16 Command line parsing on Windows

Command line parsing on Microsoft Windows now uses the standard algorithm.

The main change is the interpretation of backslash and double-quote (see the **CommandLineToArgvW** function at **http://msdn.microsoft.com/en-us/library/windows/desktop/bb776391%28v=vs.85%29.aspx**).

The command line parser sets **system:*line-arguments-list*** on startup.

15.9.17 Windows event log API

The new function **win32:record-message-in-windows-event-log** and new macro **win32:with-windows-event-log-event-source** comprise an API for recording messages in the Windows event log.

15.9.18 Rings

Ring objects can be used to hold Lisp objects and provide stack-like behavior. Each ring is limited to a maximum number of elements and can be rotated. You can control the insertion point where elements get added and removed, and iterate across the elements. For more information, see **hcl:make-ring**.

15.9.19 find-encoding-option now looks for "coding"

system:find-encoding-option now looks in the file options (EMACS-style -*- line) for an option called "coding", in addition to options called "encoding" or "external-format".

This addition is for compatibility with GNU Emacs. **system:find-encoding-option** is part of the default algorithm whereby **open** detects the encoding of a text file.

15.9.20 Process terminate methods

You can now specify the terminate method for a process. The terminate method is used by the new function **mp:process-terminate**.

See the new keywords **:remote-terminator**, **:local-terminator** and **:terminate-by-send** in **mp:process-run-function**, and also **mp:current-process-set-terminate-method**.

15.9.21 Predicate for dynamically bound symbols

The new function **hcl:symbol-dynamically-bound-p** is the predicate for whether a symbol is dynamically bound in the current environment.

15.9.22 Efficient access for 8-bit simple vectors

The new functions **system:octet-ref** and **system:base-char-ref** allow efficient access to simple vectors of element type **(unsigned-byte 8)** and strings of type **simple-base-string** in the same code.

15.9.23 Reducing size of 32-bit LispWorks when memory is low

The new function **hcl:reduce-memory** frees memory and tries to reduce the size of a 32-bit LispWorks image, without enlarging it even temporarily. It is therefore better than **hcl:clean-down** when the operating system has signalled that memory is low.

15.9.24 Timing Garbage Collector operations

Three new functions allow you to time Garbage Collector (GC) operations.

hcl:get-gc-timing reports time spent in the GC as total, user and system times, like **hcl:extended-time** except that the timing is done between your calls to **hcl:start-gc-timing** and **hcl:stop-gc-timing**.

15.9.25 Temporarily suspending profiling

After calling **hcl:start-profiling**, you can now suspend profiling by executing:

```
(stop-profiling :suspend t)
```

Profiler information already collected is retained, so if you restart profiling like this:

```
(start-profiling :initialize nil)
```

then new profiler information is added to that which was previously retained.

15.9.26 New macro dspec:replacement-source-form

Source level stepping of subforms occurring within a macro expansion works, provided that the subform is **eq** to the original subform in the call to the macro. In the rare case where a macro copies a subform, making it non-**eq**, you can use the **dspec:replacement-source-form** macro to indicate which original subform should be identified as the source code for the new form.

One example where this is helpful is **iterate**:

http://common-lisp.net/project/iterate/.

15.9.27 Representing the state of Caps Lock

The new constant **system:gesture-spec-caps-lock-bit** is used to represent the state of the Caps Lock key in a Gesture Spec where the bits are used to represent the keyboard state.

15.9.28 Test for delivered images

The predicate function **hcl:delivered-image-p** returns true when called from a delivered image, that is an image saved by a call to **lw:deliver**. It

returns **nil** otherwise. It exists in LispWorks 6.1, but is now documented in LispWorks 7.0.

15.9.29 load-data-file additional functionality and package change

hcl:load-data-file is now more flexible with respect to the file type of the loaded file; you can control whether loaded forms are evaluated; and you can provide a custom callback as an alternative to printing the result of each form. It still allows you to load data files created with previous versions of LispWorks and on most platforms.

load-data-file is now exported from the HCL package.

15.9.30 Compile and load without leaving the fasl file on disk

The **cl:compile-file** keyword argument **:load** now accepts the special value **:delete**, which means load the compiled file and then delete it (not affecting the source file). This is especially useful when using **:output-file :temp**, to avoid leaving compiled (fasl) files on disk.

15.9.31 New function socket-stream-shutdown

comm:socket-stream-shutdown performs a shutdown on one or both sides of a TCP socket connection.

15.9.32 backlog argument to TCP server functions

comm:start-up-server and **comm:accept-tcp-connections-creating-async-io-states** now accept a *backlog* argument which specifies the maximum number of pending connections for the socket in the operating system.

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.

15.10.1 New check for Cocoa Event Loop hanging

The LispWorks for Macintosh IDE now has a mechanism that checks for situations where the Cocoa Event Loop is waiting for another process, which gets an error and attempts to raise a notifier window, waiting for the Event Loop to do it, which leads to a deadlock.

When such a situation arises, LispWorks now automatically interrupts the Event Loop so that a notifier appears, the GUI updates and you can then check what went wrong.

15.10.2 Additional notifier controls

For errors in some processes, the notifier window may contain additional controls. In particular on Mac OS X the **Error handling in Cocoa event loop :** buttons allow you to control error processing in the Cocoa Event Loop (which is the process that does all the display work).

15.10.3 More fonts available in the Editor and other tools

You can now select any available font via **Preferences... > Styles > Editor Font** in the Cocoa GUI of LispWorks for Macintosh. This font is used in various LispWorks tools including the Editor.

In LispWorks 6.1 and earlier versions, you could select only fonts whose widths are (almost) integral, which is not generally guaranteed.

15.10.4 New Code Coverage tool

The Code Coverage Browser tool helps you to visualize code coverage data and is especially useful when you need to work on the data from many source files.

See “Code Coverage” on page 149 for information about the new Code Coverage interface.

15.10.5 Buffers list option in Editor

There is now an option to display a list of buffers in the Text view of Editor windows, facilitating speedy switching between buffers while editing. You can filter the buffers list in the usual way if needed.

To use the new option, select **Preferences... > Editor > General > Buffer list > Display a list of buffers in every Editor window**.

15.10.6 Preferences option to use quality drawing

Check **Preferences... > Environment > General > Use quality drawing** to make the LispWorks IDE use quality (anti-aliased) drawing for editor and graph panes. This is the default setting.

15.10.7 Listener current package preserved in Saved Sessions

The Listener tool's current package is now preserved in a Saved Session.

15.10.8 Session saving warns when Debugger or Stepper has state

Session saving now warns if there is a Debugger or Stepper tool with state. This allows you to cancel session saving and complete your debugging or stepping. If you do not want to see this warning again, and always lose the state of a Debugger or Stepper tool when saving a session, check the **Do not ask again** box in the warning dialog.

15.10.9 Setting breakpoints in the Stepper

The Stepper tool is now better at identifying Lisp definition forms with unexpected indentation. This means that you can now set breakpoints in such definitions.

In LispWorks 6.1 and earlier versions, an unhelpful message

```
No definition found
```

would appear on attempting to set a breakpoint in a definition which was not indented in the conventional Lisp way (as by the Editor command **`Indent Form`**).

15.10.10 Tool accelerator keys in KDE/Gnome emulation

Tool accelerator keys (such as **Meta+Ctrl+S** to raise the Symbol Browser tool) now work on GTK+ in KDE/Gnome editor emulation.

The full set of tool accelerators is documented in the section "Tool accelerator keys" in the *LispWorks IDE User Guide*.

15.10.11 Find window text pre-selected

Any text in the Editor tool's **Find** window is now pre-selected on all platforms when the window is raised (typically by menu command **Edit > Find...** or keystrokes **Ctrl+F** or **Command+F**).

Therefore when you want to search for other text, you can now simply type and replace the selection, without needing to take special action to remove the old text.

15.10.12 Search Files tool reports file count after failed search

When there are no matches in the Search Files tool, it displays a message which now mentions the number of files that were searched.

15.10.13 Menu command to install private patches on Windows

On Microsoft Windows the menu command **Help > Install Private Patches...** should now be used to install private patches (that is, named patch files sent to you by Lisp Support).

15.10.14 Erroneous restart removed

The restart "Edit the code where the error occurred" is no longer erroneously offered when the system does not know where the source is, such as when a buffer that has never been saved to file gives an error in the Editor.

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.

15.11.1 Improved Unicode support

The editor now supports the entire Unicode range, and provided that the system has suitable fonts it should be able to display all the characters correctly.

Note that external format **:unicode** now maps to UTF-16 rather than UCS-2. UTF-16 is quite likely to give an error trying to read a binary file. See "Unicode and other file encodings" in the *LispWorks Editor User Guide* for the recommended approach.

15.11.2 Editor bindings for keystrokes with modifiers

Use strings (rather than characters with bits, which are no longer supported) to define LispWorks Editor key sequences with modifiers, like this:

```
(editor:bind-key "Show Documentation" "Meta-Control-A")
(editor:bind-key "Search Examples" #("Control-x" "%"))
```

15.11.3 Directory mode

A buffer in Directory mode lists files and allows you to easily edit any of them, copy or move some of them to another directory, or delete some of them. It also makes it easy to keep a record of which files you already edited.

Create a Directory mode buffer by the new **List Directory** command (**Ctrl+X D** in Emacs emulation) or one of the other methods described in the guide section "Directory mode".

15.11.4 clear-undo exported and documented

The function **editor:clear-undo** is now exported and documented. It clears any undo information in an editor buffer.

In LispWorks 6.1 and earlier versions the only exported way to clear undo information was the editor command **Clear Undo**. However, this command has changed behavior as described in “Clear Undo command prompts” on page 159.

15.11.5 Clear Undo command prompts

The editor command **Clear Undo** now prompts the user for confirmation.

Programmatic use (for example as **editor:clear-undo-command** or via **capi:call-editor**) can be replaced by the function **editor:clear-undo**.

15.11.6 Un-Kill As ... commands

The new editor command **Un-Kill As String** inserts text from the kill ring as a Lisp string, surrounded by double-quotes.

The new editor command **Un-Kill As Filename** inserts the namestring of the current buffer's pathname, first converting any backslash characters to forward slash so that it does not need to be escaped in a Lisp string.

15.11.7 Save Buffer Pathname command

The new editor command **Save Buffer Pathname** pushes the namestring of the pathname of the current buffer onto the kill ring. This can then be inserted elsewhere by commands such as **Un-Kill** and **Un-Kill As Filename**.

15.11.8 Find Source For Current Package

The new editor command **Find Source For Current Package** finds the **defpackage** definition for the package at the current point. If a prefix argument is given, it first prompts for a package name.

15.11.9 New Code Coverage commands

The new commands **Code Coverage Current Buffer**, **Code Coverage File**, **Code Coverage Load Default Data** and **Code Coverage Set Default Data** allow you to visualize code coverage data in the LispWorks editor.

The functions **hcl:code-coverage-set-editor-colors** and **hcl:code-coverage-set-editor-default-data** allow you to control how the editor displays code coverage data, and which data it displays.

See "Code Coverage" on page 149 for information about the new Code Coverage interface.

15.11.10 Regular expression improvement

The replacement string used by the commands **Replace Regexp** and **Query Replace Regexp** can now contain **\d** where **d** is a digit to specify replacement by matching of **\(** and **\)** pairs.

The function **editor:regular-expression-search** can now return a vector specifying the matches of **\(** and **\)** in the pattern.

15.11.11 Editor commands now search *.cpp files

Editor commands such as **Directory Search** and **System Query Replace** now recognize files with type "cpp" as source files, and operate on them by default.

15.11.12 Editor file functions renamed

editor:append-file is now renamed as **editor:append-region-to-file** and **editor:write-file** is now **editor:write-region-to-file**.

These new names are clearer, and also avoid a clash with **lw:append-file**.

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 64-bit integer FLI types fully supported in 32-bit LispWorks

64-bit integer types such as **(:long :long)**, **:int64** and **:uint64** are now supported on 32-bit LispWorks for **fli:define-foreign-callable** and also for struct, union and pointer access with functions such as **fli:foreign-slot-value** and **fli:dereference**. In LispWorks 6.1 and earlier versions, they could only be used by **fli:define-foreign-function**.

64-bit LispWorks has always supported 64-bit integer FLI types.

15.12.2 Structure pointer allowed when passing structures by value

In LispWorks 6.1 and earlier versions, a foreign function argument whose type is a struct (that is, passing the structure by value) required a foreign struct object to be used. The LispWorks FLI now also allows a pointer to a foreign struct to be used.

15.12.3 Creating a pointer to a defined foreign symbol

The new function **fli:foreign-function-pointer** returns a FLI pointer with its address set to the address of a foreign symbol, which can be either a symbol defined in a foreign library or a foreign callable.

fli:foreign-function-pointer has several advantages over **fli:make-pointer**, when making a pointer to a defined foreign symbol. The pointer is automatically updated on image restart, for a given symbol the same pointer is returned in each call so repeated allocation is avoided. **fli:foreign-function-pointer** is especially useful for creating pointers for passing the address of foreign callables to foreign code in situations where the same address is used repeatedly.

On the other hand **fli:foreign-function-pointer** keeps a Lisp pointer to its result. Therefore if you use the FLI pointer only once you should create it with **fli:make-pointer**, to allow the Garbage Collector to discard it.

15.12.4 make-pointer requires address or symbol-name

The function **fli:make-pointer** now checks that either the *symbol-name* or *address* keyword argument is supplied, and signals an error otherwise. In LispWorks 6.1 and earlier versions it does not check, which can lead to incorrect behavior on some operating systems.

15.12.5 KOI8-R encoding supported

You can now pass and receive strings in FLI calls using the **:koi8-r** external format.

15.12.6 Incorporating a foreign module into the image with defsystem

The new **:embedded-module** member option for **lw:defsystem** creates an FLI embedded module of the given name, instead of loading the object file.

15.12.7 New type descriptor fli:released-foreign-block-pointer

The system marks foreign blocks that have been released (by **fli:foreign-block-release**) as being of this foreign type.

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 Getting information about i-dispatch methods

The new function **com:print-i-dispatch-methods** tries to get the information about the methods of an **i-dispatch** interface object, and optionally the arguments, and prints these in a readable format.

15.13.2 Reporting of failures (un)registering a server

The new function **com:set-register-server-error-reporter** allows you to control reporting of failures in calls to **DllRegisterServer** or **DllUnregisterServer**.

15.13.3 New ways to specify search paths for IDL import statements

When processing IDL import statements, **com:midl** now looks for the imported file in the directories in a list that was set by the new function **com:midl-set-import-paths** and the directories in the list returned by the new function **com:midl-default-import-paths**.

For the full description of how imported files are found, see the section "Import paths" in the *LispWorks COM/Automation User Guide and Reference Manual*.

15.13.4 New :not-specified value for optional Automation arguments

Optional Automation parameters can now be passed as the keyword value **:not-specified** if they are not needed. Alternatively, they can be omitted if all remaining optional arguments are also omitted.

15.14 Common SQL changes

15.14.1 Converting between universal time and DATE or TIMESTAMP

The functions **sql:encode-db-standard-timestamp**, **sql:encode-db-standard-date**, **sql:decode-to-db-standard-date** and **sql:decode-to-db-standard-timestamp** are now exported and documented.

They convert between Lisp universal time and standard SQL DATE and TIMESTAMP.

15.14.2 Fix for list-attribute-types with PostgreSQL and MySQL

The last element in each of the lists that **sql:list-attribute-types** returns, the *nullable* value, was wrong (not matching the documentation) for database types **:mysql** and **:postgresql**. It is fixed now to return the values as documented.

15.14.3 Warning re list-attribute-types with Access

When using ODBC to connect to Access database, the *nullable* value that **sql:list-attribute-types** returns is not reliable, at least in version 7.1. There seems to be a bug in the driver. Using ODBC with other DBMS works as documented.

15.14.4 SQL [...] syntax for = and <> generates IS NULL when used with nil

The SQL code generated for the SQL syntax **[= [field]** ***value*****]** is now **FIELD IS NULL** if *value* is **nil**.

Likewise, **[<> [field]** ***value*****]** now generates **FIELD IS NOT NULL** if *value* is **nil**.

In LispWorks 6.1 and earlier versions these generated a comparison with **NULL**, which SQL defines as always false.

15.15 Application delivery changes

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

15.15.1 New ways to keep symbols and symbol names

The new functions **hcl:deliver-keep-symbols** and **hcl:deliver-keep-symbol-names** force retention of symbols and symbol names, similarly to the **lw:deliver** keywords **:keep-symbols** and **:keep-symbol-names** but with the advantage that you can use them in your source code before loading the delivery module.

15.15.2 Retaining cross-reference information in a delivered image

The new **lw:deliver** keyword **:keep-xref-info** controls whether cross-referencing is retained in the runtime image.

Inclusion of the cross-reference functions such as **hcl:who-calls** and **hcl:calls-who** in the runtime image no longer affects whether the cross-reference information itself is retained.

15.15.3 :versioninfo no longer supports :file-os

The **lw:deliver** keyword **:versioninfo** plist no longer supports the **:file-os** key. Do not supply this.

15.15.4 LispWorks for Windows DLLs can use private assembly

You can now install a private copy of msvcr80 (Microsoft.VC80.CRT) for use by your application. It is no longer required to be in the WinSxS system folder.

Applications built with LispWorks 6.1 and earlier versions only support installation of msvcr80 by **vcredist_x86.exe** or **vcredist_x64.exe**.

15.15.5 Creating a Mac OS X application bundle.

hcl:create-macos-application-bundle is now fully documented. Please note that you should supply the argument *identifier* to provide **CFBundleIdentifier** when creating an application bundle for your own application.

15.16 CLOS/MOP changes

15.16.1 Efficient copying of slot values

The new functions **clos:copy-standard-object** and **clos:replace-standard-object** allow you to copy slot values from one CLOS instance into another (or make a new copy) very efficiently.

15.16.2 Computing the effective method function

The new MOP generic function **clos:compute-effective-method-function-from-classes** is called by LispWorks to compute the effective method function when a generic function is called with required arguments of specified types.

15.17 CLIM changes

15.17.1 Drawing with LispWorks Graphics Ports in CLIM

You can now use Graphics Ports drawing (**gp:draw-string**, **gp:draw-circle** and so on) in your CLIM application. See "Drawing with LispWorks Graphics Ports" in the *Common Lisp Interface Manager 2.0 User's Guide*.

15.18 Other changes

15.18.1 Change to default values of standard streams

The default values of the standard streams is now different. They are now set globally to synonyms to the variables **hcl:*background-output***. **hcl:*background-input*** and **hcl:*background-query-io***. The main

effect of this change is that when the LispWorks IDE is running, output to the standard output streams (**cl:*standard-output***, **cl:*trace-output*** and **cl:*error-output***) goes by default to the **mp:*background-standard-output*** in all processes. This output can therefore be viewed in the **Output** tab of Editor and Listener tools and the Output Browser tool.

In LispWorks 6.1 and earlier versions, output from processes that were not created by CAPI by default goes to **cl:*terminal-io***.

15.18.2 Implementation packages in *packages-for-warn-on-redefinition*

The list **hcl:*packages-for-warn-on-redefinition*** can now contain the keyword **:implementation**, indicating all of the packages which are part of the LispWorks implementation. This includes all of the documented packages but excludes some user packages like CL-USER and KW-USER and some packages that are used internally.

The new function **sys:package-flagged-p** allows you to query whether a package is flagged with a specified keyword such as **:implementation**.

15.18.3 process-terminate replaces process-kill

mp:process-kill is deprecated. Please use **mp:process-terminate** instead.

See also “Process terminate methods” on page 152.

15.18.4 ensure-process-cleanup changed lambda list

mp:ensure-process-cleanup has a different lambda list in LispWorks 7.0. Calls using the old lambda list are still accepted, for backwards compatibility, but please update your programs to use the new lambda list.

15.18.5 Some calls to comm:start-up-server no longer allow IPv4 connections

The behavior of *ipv6* **t** in **comm:start-up-server**, in particular whether it allows IPv4 connections, has changed.

In LispWorks 6.1 and earlier versions it depends on the default of the system.

In LispWorks 7.0 the value *ipv6* **t** means never allow IPv4 connections. The value *ipv6* **:both** can now be used to allow IPv4 connections.

15.18.6 Getting the local default IPv6 address.

The function **comm:get-local-default-ipv6-address** is new.

15.18.7 Array allocation in a specific generation deprecated

Fixnum values of the *allocation* argument of **cl:make-array** are deprecated. This is because the allocation is not actually guaranteed to be in the specified generation (although it will be in almost every call).

15.18.8 Sequence and structure predicates documented

The functions **lw:sequencep** and **lw:structurep** are now documented.

15.18.9 Changes in *features*

:lispworks7 and **:lispworks7.0** are present, **:lispworks6** and **:lispworks6.1** are 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*.

15.18.10 Loading old data files

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

15.18.11 fasl-error exported

hcl:fasl-error, the class of error signaled when loading a file which is not a proper fasl file, is now exported and documented.

15.18.12 find-throw-tag exported

hcl:find-throw-tag, the predicate for whether there is a specific catch in the dynamic scope, is now exported and documented.

15.19 Changes in the installers

15.19.1 Signing of the installer

All LispWorks installers on Microsoft Windows and Mac OS X are now signed in the name of LispWorks Ltd.

15.19.2 Dependencies in the 64-bit Linux rpm have been fixed

The 64-bit LispWorks 7.0 rpm requires **libc.so.6()(64bit)**, **libdl.so.2()(64bit)** and **libpthread.so.0()(64bit)** now. In LispWorks 6.1 and previous versions, it required **libc.so.6**, **libdl.so.2** and **libpthread.so.0**, which was wrong for Fedora and Red Hat Enterprise Linux-based distributions because those are the names of 32-bit components.

This change allows 64-bit LispWorks rpms to be installed on 64-bit computers which do not have the 32-bit subsystem.

15.20 Documentation changes

15.20.1 MOP documentation now on Help menu

The *CLOS Metaobject Protocol* specification is now available in the LispWorks IDE via the menu command **Help > Manuals > CLOS Metaobject Protocol**.

15.20.2 HTML user guides improved

The user guide parts of the LispWorks manuals are now arranged with fewer, larger HTML pages, rather than individual pages for every sub-section and sub-sub-section.

15.20.3 CAPI manuals consolidated and extended

The previously disjoint *CAPI User Guide* and *CAPI Reference Manual* are now consolidated into a combined *CAPI User Guide and Reference Manual*, with cross-references.

The new *CAPI User Guide and Reference Manual* contains two new chapters about toolbars and the self-contained examples in the LispWorks library. There are many new sections including "Focus" (for keyboard input focus), "output-pane Scrolling", "Hierarchy of panes", "Finding interfaces", "Accessing items", "Static layout", "Double list panel", "Docking layout", "Geometric queries", "Changing the graphics in the graph", "Controlling the layout" (of a graph), "Accessing the topology of the graph", "Displaying menus programmatically", "Getting the current dialog", "image-list, image-set and image-locator" and more. The section "Editor panes" is completely revised and extended.

15.20.4 Documentation improved

The COMM chapters in the *LispWorks User Guide and Reference Manual* are extended. A new section describes how *hostspec*, *service*, *local-address* and *local-port* arguments are processed. The functions **`comm:get-host-entry`** and **`comm:get-ip-default-zone-id`** are newly documented.

There is a new section in the *LispWorks User Guide and Reference Manual* "Approaching the memory limit" which describes LispWorks behavior as the image grows near to its maximum size, and describes how you can control this.

The **`:type-library`** class option to **`com:define-automation-component`** is now documented, in the LispWorks for Windows manual *LispWorks COM/Automation User Guide and Reference Manual*.

Each manual now enumerates the relevant self-contained examples in the LispWorks library along with a brief description. This helps you to find these examples via **Help > Search...**.

15.20.5 New self-contained examples

These examples are entirely new:

```
(example-edit-file "async-io/driver")

(example-edit-file "async-io/multiplication-table")

(example-edit-file "async-io/print-connection-delay")

(example-edit-file "capi/applications/price-charting")

(example-edit-file "capi/applications/rss-reader")

(example-edit-file "capi/buttons/button-panel-layout")

(example-edit-file "capi/elements/progress-bar-from-background-
thread")

(example-edit-file "capi/graphics/highlight-rectangle-pinboard")

(example-edit-file "capi/graphics/highlight-rectangle")

(example-edit-file "capi/graphics/image-scaling")

(example-edit-file "capi/graphics/plot-directly")

(example-edit-file "capi/graphics/plot-offline")

(example-edit-file "capi/graphics/tracking-pinboard-layout")

(example-edit-file "capi/output-panes/cached-display")

(example-edit-file "capi/output-panes/input-model-touch")

(example-edit-file "capi/output-panes/pane-can-scroll")

(example-edit-file "com/automation/cl-smtp/clsmtp-test")

(example-edit-file "fli/foreign-callable-example")

(example-edit-file "misc/xml-parser")
```

Also the following three examples have moved from the **examples/misc** directory to **examples/fli**

```
(example-edit-file "fli/foreign-blocks")

(example-edit-file "fli/grand-central-dispatch")

(example-edit-file "fli/invoke-foreign-block")
```

15.20.6 Newly documented functions

The functions **system:object-pointer** and **system:immediatep** are now documented. See section "Object addresses and memory" in the *LispWorks User Guide and Reference Manual* for an overview.

The function **lw:string-append*** is now documented. It creates a single string from a list of string designators. It is similar to **lw:string-append**, but takes a list as argument.

The function **system:pipe-exit-status** is now documented. **system:pid-exit-status** has been removed as it was inherently unreliable.

The generic function **system:sort-inspector-p** is now documented. It allows you to specify sorting of attributes / values in the Inspector tool in the LispWorks IDE.

15.20.7 Newly documented Editor commands

Many previously undocumented editor commands now appear in the *LispWorks Editor User Guide*.

Some are in the new section "Interaction with the GUI and the IDE", all are indexed as "Newly documented...".

15.21 Known Problems

15.21.1 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.

15.21.2 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/**.

15.21.3 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.

15.21.4 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).

15.22 Binary Incompatibility

If you have binaries (fasl files) which were compiled using LispWorks 6.1 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 7.0.

Index

A

accept-tcp-connections-creating-async-io-states function 154
Access 163
accessor functions
  editor-pane-text 146
  interactive-pane-stream 146
  interactive-pane-top-level-function 146
  interface-override-cursor 171, 173
anti-aliased drawing in the IDE 156
antialiased drawing in the IDE 156
append-file function 160
append-region-to-file function 160
Asynchronous I/O 149

B

background-input variable 165
background-output variable 165
background-query-io variable 165
base-char-ref function 152
base-string type 149
bitmap-port class 145
bmp-char type 148
bmp-string type 148
browser-pane class 138
browser-pane-available-p function 138
building-universal-intermediate-p function 134

C

calls-who function 164
character type 148
char-bit function 148
char-bits function 148
char-bits-limit constant 148
char-code-limit constant 147
char-font function 148
char-font-limit constant 148
classes
  bitmap-port 145
  browser-pane 138
  docking-layout 173
  editor-pane 136, 142, 143
  graphics-port-mixin 144
  graph-pane 143
  grid-layout 136
  interactive-pane 146
  interface 139
  layout 136, 141
  metafile-port 144
  multi-column-list-pane 141
  multi-column-list-panel 137, 173
  output-pane 135, 136, 138, 141, 143, 173
  pinboard-layout 137, 138
  printer-port 144
  sorted-object 137
  text-input-pane 142, 171, 173

titled-object 144
title-pane 144
toolbar 139
toolbar-component 139
tree-view 141, 173
clean-down function 152
Clear Undo editor command 158, 159
clear-undo function 158
clear-undo-command function 159
Code Coverage Current Buffer editor command 159
Code Coverage File editor command 159
Code Coverage Load Default Data editor command 159
Code Coverage Set Default Data editor command 159
code-char function 147
code-coverage-set-editor-colors function 159
code-coverage-set-editor-default-data function 159
coerce-to-gesture-spec function 148
collection-find-string function 145
collection-search function 145
color-from-premultiplied function 142
color-to-premultiplied function 142
command line arguments
-init 127
-init on Linux 91, 94
-init on Mac OS X 74, 77
-init on Unix 106, 107, 108
-init on Windows 83, 86
quoting 151
-siteinit on Linux 91
-siteinit on Mac OS X 74
-siteinit on Unix 106, 107
-siteinit on Windows 83
Command line parsing on Windows 151
compile-file function 154
compute-effective-method-function-from-classes generic function 165
condition classes
fasl-error 167
constants
char-bits-limit 148
char-code-limit 147
char-font-limit 148
gesture-spec-caps-lock-bit 153
contain function 140
convert-to-screen function 138
copy-standard-object function 165
corrupted executable 113
create-bitmap-port function 145
create-dummy-graphics-port function 138
create-macos-application-bundle function 165
create-universal-binary function 134
current-process-send function 134
current-process-set-terminate-method function 152

D

decode-to-db-standard-date function 163
decode-to-db-standard-timestamp function 163
define-automation-component macro 169
defsystem macro 162
deliver function 164
delivered-image-p function 153
deliver-keep-symbol-names function 164
deliver-keep-symbols function 164
destroy-dependent-object generic function 138
directory function 150
Directory Search editor command 160
display-dialog function 173
display-non-focus-message function 139
display-tooltip function 172
dither-color-spec function 145
docking-layout class 173
draw-circle function 165
draw-image function 143
draw-pinboard-layout-objects function 137
draw-string function 165
dump-forms-to-file function 167

E

editor commands

Clear Undo 158, 159
Code Coverage Current Buffer 159
Code Coverage File 159
Code Coverage Load Default Data 159
Code Coverage Set Default Data 159
Directory Search 160
Find Source For Current Package 159
List Directory 158
Query Replace Regexp 160
Replace Regexp 160
Save Buffer Pathname 159
Select Buffer 142
System Query Replace 160
Un-Kill As Filename 159
Un-Kill As String 159
editor font size
on Macintosh 136, 155
editor fonts
on Macintosh 136, 155
editor-pane class 136, 142, 143, 171
editor-pane-text accessor function 146
encode-db-standard-date function 163
encode-db-standard-timestamp function 163
ensure-process-cleanup function 166
enter-debugger-directly variable 172
error-output variable 166
errors while building application 112
errors while delivering application 112
Event-driven socket streams 149
extended-time macro 125, 153

F

Failed to enlarge memory 113
fasl-error condition class 167
fast-directory-files function 150
features variable 132, 167
File > Close menu command 139
File > Exit menu command 139
fill-style graphics state parameter 173
Find Source For Current Package editor command 159
Find... menu command 157
find-encoding-option function 152
find-matching-fonts function 143
find-regexp-in-string function 150
find-throw-tag function 168
:flag initarg 142
FLI type descriptors
released-foreign-block-pointer 162
font-dual-width-p function 145
font-single-width-p function 145
foreign-block-release function 162
foreign-function-pointer function 161
funcall-async function 150
funcall-async-list function 150
functions
accept-tcp-connections-creating-async-io-states 154
append-file 160
append-region-to-file 160
base-char-ref 152
browser-pane-available-p 138
building-universal-intermediate-p 134
calls-who 164
char-bit 148
char-bits 148
char-font 148
clean-down 152
clear-undo 158
clear-undo-command 159
code-char 147
code-coverage-set-editor-colors 159
code-coverage-set-editor-default-data 159
coerce-to-gesture-spec 148
collection-find-string 145
collection-search 145
color-from-premultiplied 142
color-to-premultiplied 142
compile-file 154
contain 140
convert-to-screen 138
copy-standard-object 165
create-bitmap-port 145
create-dummy-graphics-port 138
create-macos-application-bundle 165
create-universal-binary 134

current-process-send 134
current-process-set-terminate-method 152
decode-to-db-standard-date 163
decode-to-db-standard-timestamp 163
deliver 164
delivered-image-p 153
deliver-keep-symbol-names 164
deliver-keep-symbols 164
directory 150
display-dialog 173
display-non-focus-message 139
display-tooltip 172
dither-color-spec 145
draw-circle 165
draw-image 143
draw-pinboard-layout-objects 137
draw-string 165
dump-forms-to-file 167
encode-db-standard-date 163
encode-db-standard-timestamp 163
ensure-process-cleanup 166
fast-directory-files 150
find-encoding-option 152
find-matching-fonts 143
find-regexp-in-string 150
find-throw-tag 168
font-dual-width-p 145
font-single-width-p 145
foreign-block-release 162
foreign-function-pointer 161
funcall-async 150
funcall-async-list 150
gesture-spec-to-character 148
get-gc-timing 153
get-host-entry 169
get-ip-default-zone-id 169
get-local-default-ipv6-address 167
immediatep 171
initialize-dithers 145
invalidate-rectangle-from-points 141
list-attribute-types 163
load-data-file 154, 167
log-bug-form 128
mailbox-wait 134
make-array 167
make-dither 145
make-gesture-spec 148
make-pointer 161
make-ring 151
make-sorting-description 137
midl 162
midl-default-import-paths 162
midl-set-import-paths 162
modify-editor-pane-buffer 142
modify-multi-column-list-panel-columns 137
object-pointer 171
octet-ref 152
output-pane-stop-composition 140
package-flagged-p 166
pane-can-restore-display-p 139
pane-close-display 145
pane-modifiers-state 138
pane-restore-display 139
pipe-exit-status 151, 171
popup-menu-force-popdown 139
print-file 142
print-i-dispatch-methods 162
print-text 142
process-kill 166
process-run-function 152
process-terminate 152, 166
prompt-for-files 140
prompt-with-list-non-focus 140
read-external-image 142
record-dependent-object 138
record-message-in-windows-event-log 151
reduce-memory 152
regular-expression-search 160
replace-standard-object 165
room 125
run-shell-command 151
save-argument-real-p 134
save-universal-from-script 134
sequencep 167
set-register-server-error-

reporter 162
socket-stream-shutdown 154
start-environment 15, 108
start-gc-timing 153
start-up-server 154, 166
stop-gc-timing 153
string-append 171
string-append* 171
structurep 167
symbol-dynamically-bound-p 152
unicode-alpha-char-p 149
unicode-char-equal 149
unicode-string-equal 149
unicode-string-greaterp 149
unicode-string-not-lessp 149
unrecord-dependent-object 138
who-calls 164
write-file 160
write-region-to-file 160

G

Garbage Collector message 113
Garbage Collector output 113
GC message 113
GC output 113
generic functions
compute-effective-method-function-from-classes 165
destroy-dependent-object 138
output-pane-resize 137
pane-adjusted-offset 145
pane-adjusted-position 145
pane-string 146
pinboard-layout-display 137
pinboard-pane-position 145
pinboard-pane-size 145
sort-inspector-p 171
static-layout-child-position 145
static-layout-child-size 145
gesture-spec-caps-lock-bit constant 153
gesture-spec-to-character function 148
get-gc-timing function 153
get-host-entry function 169
get-ip-default-zone-id function 169
get-local-default-ipv6-address function 167
graphics-port-mixin class 144
graph-pane class 143
grid-layout class 136
GTK 135
GTK+ 135

H

:help-callback initarg 173

I

IDE 154
immediatep function 171
initialize-dithers function 145
Install Private Patches... menu command 114, 124
Integrated Development Environment 154
interactive-pane class 146
interactive-pane-stream accessor function 146
interactive-pane-top-level-function accessor function 146
interactive-stream 146
interactive-stream-stream 146
interactive-stream-top-level-function 146
interface class 139
interface-override-cursor accessor function 171, 173
invalidate-rectangle-from-points function 141

K

:keep-symbol-names delivery keyword 164
:keep-symbols delivery keyword 164
:keep-xref-info delivery keyword 164

L

layout class 136, 141
line-arguments-list variable 151
LispWorks fails to start 113
LispWorks for Android Runtime 69
LispWorks for iOS Runtime 69
LispWorks for Mobile Runtime 69
LispWorks IDE tools
Debugger 156
Editor 157
Listener 156
Notifier 155
Search Files 157

Stepper 156
List Directory 158
list-attribute-types function 163
load-data-file function 154, 167
log-bug-form function 128

M

macros
define-automation-component 169
defsystem 162
extended-time 125, 153
profile 125
replacement-source-form 153
with-bitmap-port 145
with-dither 145
with-output-to-fasl-file 167
with-page 173
with-print-job 144
with-windows-event-log-event-source 151
mailbox type 150
mailbox-wait function 134
make-array function 167
make-dither function 145
make-gesture-spec function 148
make-pointer function 161
make-ring function 151
make-sorting-description function 137
metafile-port class 144
midl function 162
midl-default-import-paths function 162
midl-set-import-paths function 162
:min-column-width initarg 136
:min-row-height initarg 136
modify-editor-pane-buffer function 142
modify-multi-column-list-panel-columns function 137
Motif 135
move LispWorks to another computer 129
moving LispWorks to another computer 129
multi-column-list-pane class 141
multi-column-list-panel class 137, 173
multi-touch gestures 135
MySQL 163

N

"Not yet multiprocessing." error 112

O

object-pointer function 171
octet-ref function 152
operation graphics state parameter 173
option-pane class 171
output-pane class 135, 136, 138, 141, 143, 173
output-pane-resize generic function 137
output-pane-stop-composition function 140

P

package-flagged-p function 166
packages-for-warn-on-redefinition variable 166
pane-adjusted-offset generic function 145
pane-adjusted-position generic function 145
pane-can-restore-display-p function 139
pane-close-display function 145
pane-modifiers-state function 138
pane-restore-display function 139
pane-string generic function 146
pattern graphics state parameter 173
pinboard-layout class 137, 138
pinboard-layout-display generic function 137
pinboard-pane-position generic function 145
pinboard-pane-size generic function 145
pipe-exit-status function 151, 171
poor performance 125
popup-menu-force-popdown function 139
port-owner function 141
PostgreSQL 163
Preferences... menu command 155
printer-port class 144
print-file function 142
print-i-dispatch-methods function 162
print-text function 142
private patches
not loaded on Windows 114

process-kill function 166
process-run-function function 152
process-terminate function 152, 166
profile macro 125
prompt-for-files function 140
prompt-with-list-non-focus function 140

Q

Query Replace Regexp 160

R

read-external-image function 142
record-dependent-object function 138
record-message-in-windows-event-log function 151
reduce-memory function 152
Register... menu command 15, 20, 34, 42, 50, 59
regular-expression-search function 160
released-foreign-block-pointer FLI type descriptor 162
Replace Regexp 160
replacement-source-form macro 153
replace-standard-object function 165
room function 125
run-shell-command function 151

S

Save Buffer Pathname editor command 159
save-argument-real-p function 134
save-universal-from-script function 134
Select Buffer editor command 142
sequencep function 167
set-register-server-error-reporter function 162
simple-bmp-string type 148
simple-char type 148
simple-text-string type 148
socket-stream-shutdown function 154
sorted-object class 137
sort-inspector-p generic function 171
standard-output variable 166
start-environment function 15, 108
start-gc-timing function 153
start-up-server function 154, 166
static-layout-child-position generic function 145
static-layout-child-size generic function 145
stipple graphics state parameter 173
stop-gc-timing function 153
string-append function 171
string-append* function 171
structurep function 167
symbol-dynamically-bound-p function 152
System Query Replace editor command 160

T

text-input-pane class 142, 171, 173
text-string type 148
titled-object class 144
title-pane class 144
:title-position initarg 144
toolbar class 139
toolbar-component class 139
touchscreen gestures 135
trace-output variable 166
trackpad gestures 135
transfer LispWorks to another computer 129
transferring LispWorks to another computer 129
tree-view class 141, 173
types
  base-string 149
  bmp-char 148
  bmp-string 148
  character 148
  mailbox 150
  simple-bmp-string 148
  simple-char 148
  simple-text-string 148
  text-string 148
  unlocked-queue 150

U

unicode-alpha-char-p function 149
unicode-char-equal function 149
unicode-string-equal function 149
unicode-string-greaterp func-

tion 149
unicode-string-not-lessp function 149
uninstalling LispWorks
on AIX 59
on FreeBSD 50
on Linux 33
on Macintosh 15
on Windows 20
on x86/x64 Solaris 41
universal binaries
not supported 133
universal binary
not supported 133
Un-Kill As Filename editor command 159
Un-Kill As String editor command 159
unlocked-queue type 150
unrecord-dependent-object function 138
update-screen-interfaces-hooks variable 146

V

variables
background-input 165
background-output 165
background-query-io 165
enter-debugger-directly 172
error-output 166
features 132, 167
line-arguments-list 151
packages-for-warn-on-redefinition 166
standard-output 166
trace-output 166
update-screen-interfaces-hooks 146
:visible-border initarg 173

W

who-calls function 164
window system 135
Windows event log 151
with-bitmap-port macro 145
with-dither macro 145
with-output-to-fasl-file macro 167
with-page macro 173
with-print-job macro 144
with-windows-event-log-event-source macro 151
write-file function 160
write-region-to-file function 160