LispWorks®

Release Notes and Installation Guide

Version 5.1

Copyright and Trademarks

LispWorks Release Notes and Installation Guide

Version 5.1

March 2008

Copyright © 2008 by LispWorks Ltd.

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of LispWorks Ltd.

The information in this publication is provided for information only, is subject to change without notice, and should not be construed as a commitment by LispWorks Ltd. LispWorks Ltd assumes no responsibility or liability for any errors or inaccuracies that may appear in this publication. The software described in this book is furnished under license and may only be used or copied in accordance with the terms of that license.

LispWorks and KnowledgeWorks are registered trademarks of LispWorks Ltd.

Adobe and PostScript are registered trademarks of Adobe Systems Incorporated. Other brand or product names are the registered trademarks or trademarks of their respective holders.

The code for walker.lisp and compute-combination-points is excerpted with permission from PCL, Copyright © 1985, 1986, 1987, 1988 Xerox Corporation.

The XP Pretty Printer bears the following copyright notice, which applies to the parts of LispWorks derived therefrom:
Copyright © 1989 by the Massachusetts Institute of Technology, Cambridge, Massachusetts.
Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that this copyright and permission notice appear in all copies and supporting documentation, and that the name of M.I.T. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T. makes no representation about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty. M.I.T. disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness. In no event shall M.I.T. be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

LispWorks contains part of ICU software obtained from http://source.icu-project.org and which bears the following copyright and permission notice:
ICU License - ICU 1.8.1 and later
COPYRIGHT AND PERMISSION NOTICE
Copyright © 1995-2006 International Business Machines Corporation and others. All rights reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder. All trademarks and registered trademarks mentioned herein are the property of their respective owners.

US Government Restricted Rights
The LispWorks Software is a commercial computer software program developed at private expense and is provided with restricted rights. The LispWorks Software may not be used, reproduced, or disclosed by the Government except as set forth in the accompanying End User License Agreement and as provided in DFARS 227.7202-1(a), 227.7202-3(a) (1995), FAR 12.212(a)(1995), FAR 52.227-19, and/or FAR 52.227-14 Alt III, as applicable. Rights reserved under the copyright laws of the United States.

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: 305 468 5262

From elsewhere: +44 870 2206189

www.lispworks.com

Contents

1 Introduction 1

LispWorks Editions 1
LispWorks for UNIX 2
Further details 3
About this Guide 3

2 Installation on Mac OS X 5

Choosing the Graphical User Interface 5
Documentation 5
Software and hardware requirements 6
Installing LispWorks for Macintosh 7
Starting LispWorks for Macintosh 11
Upgrading to LispWorks Enterprise Edition 13

3 Installation on Windows 15

Documentation 15
Installing LispWorks for Windows 16
Upgrading to LispWorks Enterprise Edition 18

4 Installation on Linux 19

Software and hardware requirements 19
License agreement 21
Software on the CD-ROM 21
Installing LispWorks for Linux 22
LispWorks looks for a license key 28
Running LispWorks 28
Configuring the image 29
Printable LispWorks documentation 29
Uninstalling LispWorks for Linux 29
Upgrading to LispWorks Enterprise Edition 30

5 Installation on FreeBSD 31

Software and hardware requirements 31
License agreement 32
Software on the CD-ROM 33
Installing LispWorks for FreeBSD 33
LispWorks looks for a license key 35
Running LispWorks 36
Configuring the image 37
Printable LispWorks documentation 37
Uninstalling LispWorks for FreeBSD 37
Upgrading to LispWorks Enterprise Edition 37

6 Installation on UNIX 39

Introduction 39
Extracting software from the CD-ROM 39
Moving the LispWorks image and library 41
Obtaining and Installing your license keys 42
Configuring the LispWorks image 43
Using the Documentation 45
Using Layered Products on HP PA or Sun Sparc (32-bit) 45

7 Configuration Details on Mac OS X 47

Introduction 47

License keys 48
Configuring your LispWorks installation 48
Saving and testing the configured image 50
Initializing LispWorks 52
Loading CLIM 2.0 53
The Common SQL interface 54
Common Prolog and KnowledgeWorks 56

8 Configuration Details on Windows 57

Introduction 57
License keys 58
Configuring your LispWorks installation 58
Saving and testing the configured image 59
Initializing LispWorks 61
Loading CLIM 2.0 62
The Common SQL interface 63
Common Prolog and KnowledgeWorks 63

9 Configuration Details on Linux and FreeBSD 65

Introduction 65
License keys 66
Configuring your LispWorks installation 66
Saving and testing the configured image 68
Initializing LispWorks 69
Loading CLIM 2.0 70
The Common SQL interface 71
Common Prolog and KnowledgeWorks 72
Documentation for LispWorks for FreeBSD 72

10 Configuration Details on UNIX 73

Disk requirements 73
Software Requirements 73
The CD-ROM 73
Installing LispWorks 75
Components of the LispWorks distribution 79

Printing copies of the LispWorks documentation 81
Configuring your LispWorks installation 81
LispWorks initialization arguments 85

11 Troubleshooting, Patches and Reporting Bugs 87

Troubleshooting 87
Troubleshooting on Mac OS X 89
Troubleshooting on Linux 90
Troubleshooting on FreeBSD 92
Troubleshooting on UNIX 93
Troubleshooting on X11/Motif 94
Updating with patches 96
Reporting bugs 98

12 Release Notes 105

Additional support for 64-bit on Macintosh 105
Running on 64-bit machines 106
New CAPI features 106
Other CAPI changes 110
New graphics ports features 110
More new features 111
IDE changes 115
Editor changes 120
Foreign Language interface changes 123
COM/Automation changes 125
Common SQL changes 125
Application delivery changes 126
CLOS/MOP changes 127
CLIM changes 128
Other changes 128
Documentation changes 134
Binary Incompatibilty 134
Known Problems 134
Recyclable packaging 137

Index 139

1 Introduction

1.1 LispWorks Editions

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

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

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

1.1.1 Personal Edition

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

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

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

- A heap size limit
- A time limit of 5 hours for each session.
- The functions **save-image**, **deliver**, and **load-all-patches** are not available.
- Initialization files are not available.
- Professional and Enterprise Edition module loading is not included.

LispWorks 5.1 Personal Edition has no license fee. Download it from

www.lispworks.com/downloads.

1.1.2 Professional Edition

LispWorks 5.1 Professional Edition includes:

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

1.1.3 Enterprise Edition

LispWorks 5.1 Enterprise Edition provides further support for the software needs of the modern enterprise, including:

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

1.2 LispWorks for UNIX

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

LispWorks for UNIX 5.1 is available with a basic developer license, and the add-on products CLIM, KnowledgeWorks, LispWorks ORB and Application Delivery are each separately available.

1.3 Further details

For further information about LispWorks products visit

www.lispworks.com

To purchase LispWorks please follow the instructions at:

www.lispworks.com/buy

1.4 About this Guide

This document is an installation guide and release notes for LispWorks 5.1 on Mac OS X, Windows, Linux, FreeBSD and UNIX platforms. It also explains how to configure LispWorks to best suit your local conditions and needs.

This guide provides instructions for installing and loading the modules included with each Edition or add-on product.

1.4.1 Installation and Configuration

The next four chapters explain in brief and sufficient terms how to complete a LispWorks installation on Mac OS X, Windows, Linux or UNIX. Choose the chapter for your platform: Chapter 2, “Installation on Mac OS X”, Chapter 3, “Installation on Windows” or Chapter 4, “Installation on Linux” or Chapter 6, “Installation on UNIX”.

The following four chapters explain in detail everything necessary to configure, run, and test LispWorks 5.1. Choose the chapter for your platform: Chapter 7, “Configuration Details on Mac OS X”, Chapter 8, “Configuration Details on Windows”, Chapter 9, “Configuration Details on Linux and FreeBSD” or Chapter 10, “Configuration Details on UNIX”. This also includes sections on initializing LispWorks and loading some of the modules. You should have no difficulty configuring, running, and testing LispWorks using these instructions if you have a basic familiarity with your operating system and Common Lisp.

1.4.2 Troubleshooting

Chapter 11, “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 12, “Release Notes”, highlights what is new in this release and special issues for the user’s consideration.

2

Installation on Mac OS X

This chapter is an installation guide for LispWorks 5.1 for Macintosh. Chapter 7 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

2.1 Choosing the Graphical User Interface

LispWorks for Macintosh supports two different graphical interfaces. Most users choose the native Mac OS X GUI, but you can use the Motif GUI instead.

Different executables and supporting files are supplied for the two options. You need to decide at installation time which of the GUIs you will be using, or decide to install support for both. If you install just one GUI option and later decide to install the other GUI option, you can simply run the installer again.

LispWorks for Macintosh Personal Edition supports only the native Mac OS X GUI.

2.2 Documentation

The LispWorks documentation set is included in two electronic forms: HTML and PDF. You can chose whether to install it as described in Section 2.4, “Installing LispWorks for Macintosh”.

The HTML version can be used from within the LispWorks environment via the **Help** menu. You will need a suitable web browser installed. You can also reach the HTML documentation at the page **`manual/online/intro.htm`** in the LispWorks library. If you choose not to install the documentation, you will not be able to access the HTML Documentation from the LispWorks **Help** menu.

The PDF version is suitable for printing. Each manual in the documentation set is presented in a separate PDF file in the LispWorks library under **`manual/offline/pdf`**. To view and print these files, you will need a PDF viewer such as Adobe® Reader®. This can be downloaded from the Adobe website at **`www.adobe.com`**.

2.3 Software and hardware requirements

LispWorks 5.1 is a universal binary, which supports Macintosh computers containing either PowerPC or Intel CPUs.

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

Product	Hardware Requirements	Software Requirements
LispWorks (32-bit) for Macintosh	Intel or G3/G4/G5 processor. 32MB of memory, preferably 64MB. 200MB of disk space including documentation.	Mac OS X version 10.3.x, 10.4.x or 10.5.x. OpenMotif 2.3 if you want to run the X11/Motif GUI.
LispWorks (64-bit) for Macintosh	Intel or G5 processor. 64MB of memory, preferably 128MB. 140MB of disk space including documentation	Mac OS X version 10.5.x. OpenMotif 2.3 if you want to run the X11/Motif GUI.

Table 2.1

2.4 Installing LispWorks for Macintosh

2.4.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as an installer containing version 5.1. There may be a downloadable patch bundle which upgrades LispWorks to version 5.1.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 5.1.

2.4.2 Information for Beta testers

Users of LispWorks 5.1 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 5.1. You can run the Beta installer and select the **Uninstall** option (and then remove any patches) or simply drag the **`LispWorks 5.1`** folder to the trash.

2.4.3 Information for users of previous versions

You can install LispWorks 5.1 in the same location as LispWorks 5.0 or previous versions. If you always choose the default install location, a new **`LispWorks 5.1`** folder will be created alongside the other versions.

Similarly LispWorks Personal Edition 5.1 can be installed in the same location as previous versions.

2.4.4 Use an adminstrator account

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

However, a non-administrator may install LispWorks elsewhere.

2.4.5 Launch the LispWorks installer

If you have downloaded LispWorks, you may need to mount the disk image containing the installer. This is called **`LispWorks-5.1.dmg`** or **`LispWorks64bit-5.1.dmg`** — simply double-click on the **`.dmg`** file to mount it.

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

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

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

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

2.4.6 The Read Me

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

2.4.7 The License Agreement

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

2.4.8 Select Destination

All the files installed with LispWorks are placed in the LispWorks folder, which is named **LispWorks 5.1**, **LispWorks 5.1 (64-bit)** or **LispWorks Personal 5.1** depending on which edition you are installing. By default, the LispWorks folder is placed in the main **Applications** folder but you can choose an alternative location during installation by clicking the **Select Folder...** button.

Click **Continue** after selecting a folder.

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

2.4.9 Choose your installation type

2.4.9.1 The native Mac OS X GUI

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

2.4.9.2 The X11/Motif GUI

If you want to install LispWorks with the X11/Motif GUI, choose **Custom Install** and select the option "LispWorks with X11/Motif IDE".

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

- An X server such as Apple's X11.app, available at **`www.apple.com`**.
- OpenMotif 2.3. For availability see "Obtaining OpenMotif" on page 10.

Neither X11 or Motif are required at the time you install LispWorks, however.

The X11/Motif GUI is not available for the Personal Edition.

2.4.9.3 The Documentation

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

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

2.4.10 Installing

Now click **Install**.

2.4.11 Enter License Data

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

If you have received LispWorks on a CD-ROM then your license key is supplied on a label inside the CD-ROM case. If you have downloaded the product then 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.

2.4.12 Add LispWorks to the Dock

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

Note: On Mac OS X 10.4 and 10.5, LispWorks may not be visible in the Dock until you restart the computer or log out and then log back in.

2.4.13 Finishing up

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

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

2.4.14 Installing Patches

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

2.4.15 Obtaining OpenMotif

LispWorks 5.1 for Macintosh on X11/Motif requires Open Motif 2.3.

The library for 32-bit LispWorks is **/usr/local/lib/libXm.4.dylib**. You can can build a PowerPC or Intel binary from the sources at **www.motifzone.net**.

Apple do not currently support a 64-bit X11 installation. Contact Lisp Support if you need a library suitable for 64-bit LispWorks.

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 8.

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-5-1-0-macos-universal`** image.

See Section 7.3, "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-5-1-0-macos-universal`**, and **`save-image`** is not available.

2.5.2 Start the X11/Motif LispWorks GUI

Assuming you have installed this option, and that you have X11 running and Motif installed, you can now start LispWorks with the X11/Motif GUI.

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

1. Call the function **`env:start-environment`**

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

```
xterm% cd "/Applications/LispWorks 5.1"
xterm% ./lispworks-5-1-0-macos-universal-motif
LispWorks(R): The Common Lisp Programming Environment
Copyright (C) 1987-2008 LispWorks Ltd.  All rights reserved.
Version 5.1.0
Saved by LispWorks as lispworks-5-1-0-darwin-motif, at 29 Feb
2008 14:37
User dubya on octane
; Loading text file /Applications/LispWorks 5.1/Library/lib/5-1-
0-0/config/siteinit.lisp
; Loading text file /Applications/LispWorks 5.1/Library/lib/5-1-
0-0/private-patches/load.lisp
; Loading text file /u/ldisk/dubya/.lispworks

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

The LispWorks X11/Motif IDE and Lisp Monitor window should appear.

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

2. Pass the **-env** command line argument

   The **-env** command line argument causes the function **env:start-environment** to be called.

   Follow this session in the X11 terminal:

```
xterm% cd "/Applications/LispWorks 5.1"
xterm% ./lispworks-5-1-0-macos-universal-motif -env
```

   The LispWorks X11/Motif IDE and Lisp Monitor window should appear.

3. Create an image which starts the GUI automatically

   If you want to create a LispWorks image which starts the GUI automatically, you should make a configuration script that calls

   (save-image ... :environment t)

   and pass it to the supplied **lispworks-5-1-0-macos-universal-motif** image. Note: This will create a non-universal binary, containing only the architecture on which you call **save-image**.

See Section 7.3, "Configuring your LispWorks installation" for more information about configuring your LispWorks image for your own needs.

2.6 Upgrading to LispWorks Enterprise Edition

You can upgrade from LispWorks Professional Edition by **Help > Register...** and enter an Enterprise license key.

3

Installation on Windows

This chapter is an installation guide for LispWorks 5.1 for Windows and LispWorks 5.1 (64-bit) for Windows. Chapter 8 discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

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 Common LispWorks environment via the **Help** menu. It is also available from the **Start** menu under **Start > All Programs > LispWorks 5.1 > HTML Documentation**.

The PDF version is suitable for printing. Each manual in the documentation set is presented in a separate PDF file, available from the **Start** menu under **Start > All Programs > LispWorks 5.1 > PDF Documentation**. To view and print these files, you will need a PDF viewer such as Adobe® Reader®. If you do not already have this, it can be downloaded from the Adobe website.

3.2 Installing LispWorks for Windows

3.2.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as an installer containing version 5.1. There may be a downloadable patch bundle which upgrades LispWorks to version 5.1.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 5.1.

3.2.2 Visual Studio runtime components and Windows Installer

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

It will also automatically install Windows Installer 3.1 when needed (for example on Windows 2000).

3.2.3 Installing over previous versions

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

You can also install LispWorks 5.1 without uninstalling older versions such as Xanalys LispWorks 4.4 or Xanalys LispWorks 4.3 provided that the chosen installation directory is different.

The LispWorks Personal Edition installation behaves in the same way.

3.2.4 Information for Beta testers

Users of LispWorks 5.1 Beta should completely uninstall it before installing LispWorks 5.1. Remember to remove any patches added since the initial beta release.

3.2.5 To install LispWorks

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

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

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

3.2.5.1 Entering the License Data

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

If you have received LispWorks on a CD-ROM then your license key is supplied on a label inside the CD-ROM case. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

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

3.2.5.2 Installation location

By default LispWorks installs in all users space in
C:\Program Files\LispWorks

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

3.2.5.3 Installing the Documentation

By default all the documentation is installed.

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

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

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

3.2.5.4 Installing Patches

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

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

3.2.5.5 Starting LispWorks

When the installation is complete, you can start LispWorks by choosing **Start > All Programs > LispWorks 5.1 > LispWorks**.

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

3.3 Upgrading to LispWorks Enterprise Edition

You can upgrade from LispWorks Professional Edition by **Help > Register...** and enter an Enterprise license key.

4

Installation on Linux

This chapter is an installation guide for LispWorks 5.1 for Linux and LispWorks 5.1 (64-bit) for Linux. Chapter 9, discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

4.1 Software and hardware requirements

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

Hardware Requirements	Software Requirements
32MB of memory, preferably 64MB	RedHat Linux (version 9 or later) or a distribution with kernel version 2.4 or later that supports NPTL and glibc 2.3.2 or later.

Table 4.1

Hardware Requirements	Software Requirements
155MB of disk space for Enterprise Edition (32-bit) plus documentation	OpenMotif 2.2.x
175MB of disk space for Enterprise Edition (64-bit) plus documentation	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 4.1

4.1.1 Motif libraries

LispWorks 5.1 for Linux requires that the X11 release 6 (or higher) and OpenMotif (version 2.2 or higher) are installed on your machine. 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.

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

Note: You should be able to run LispWorks 5.1 and LispWorks 5.0 simultaneously with OpenMotif installed.

4.1.2 Disk requirements during installation

LispWorks requires about 30MB for 32-bit and 50MB for 64-bit to install without documentation and optional modules. 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 175MB.

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

4.2 License agreement

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

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

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

Now run the one of following scripts.

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

- For the Professional and Enterprise Editions, run

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

- For the Personal Edition, run:

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

Enter "yes" if you agree to the license terms.

4.3 Software on the CD-ROM

LispWorks 5.1 for Linux is supplied on a CD-ROM in two different formats: RedHat Package Management (RPM) files and **tar** files. RPM is a utility like **tar**, except it can actually install products after unpacking them. See Section 4.4.3 for more information. Both formats are in the **x86-linux** and **amd64-linux** directories on your CD-ROM.

4.3.1 Professional and Enterprise Edition distributions

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

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

For the Professional Edition the separately installable packages are:

```
lispworks-clim
```

and for the Enterprise Edition the separately installable packages are:

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

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

The package name for the Personal Edition is **lispworks-personal**.

4.4 Installing LispWorks for Linux

4.4.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as an installer containing version 5.1. There may be a downloadable patch bundle which upgrades LispWorks to version 5.1.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

LispWorks Personal Edition is supplied as an installer containing version 5.1.

4.4.2 Information for Beta testers

Users of LispWorks 5.1 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 5.1.

See “Uninstalling LispWorks for Linux” on page 29 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 5.1 Beta installed, please uninstall it before installing this product. See Section 4.9, “Uninstalling LispWorks for Linux”.

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

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

try upgrading to RPM 3.0.2 or greater.

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

4.4.3.1 Installation directories

By default 32-bit LispWorks is installed in **/usr/lib/LispWorks** and a symbolic link to the executable is placed in **/usr/bin/lispworks-5-1-0-x86-linux**. Similarly, 64-bit LispWorks is installed in **/usr/lib64/LispWorks** and a symbolic link to the executable is placed in **/usr/bin/lispworks-5-1-0-amd64-linux**. However, the RPM is relocatable, and the **--prefix** option can be used to allow the installation of LispWorks in a non-default directory. The default prefix is **/usr**.

Note: RPM version 4.2 has 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 91 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-5.1-*n***.***arch***.rpm**

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

Note: For the Personal Edition, use **lispworks-personal-5.1-*.i386.rpm** wherever **lispworks-5.1-*.i386.rpm** is mentioned in this document. See Section 1.1.1, "Personal Edition" for more information specific to the Personal Edition.

4.4.3.3 Installing or upgrading LispWorks for Linux

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

1. Locate the RPM installation file **lispworks-5.1-*n*.i386.rpm**.
2. Install or upgrade LispWorks in the standard RPM way, for example:

   rpm --install lispworks-5.1-*n*.i386.rpm

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

   rpm --install --prefix <*directory*> lispworks-5.1-*n*.i386.rpm

   installs LispWorks in **<*directory*>**.

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

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

See Section 4.6 for instructions on entering your license details.

4.4.3.4 Installing CLIM 2.0

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

File Distribution	Layered Product
lispworks-clim-5.1-*n*.i386.rpm	CLIM 2.0

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

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

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

4.4.3.5 Installing loadable Enterprise Edition modules

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

File Distribution	Layered Product
lispworks-clim-5.1-*n*.i386.rpm	CLIM 2.0
lispworks-kw-5.1-*n*.i386.rpm	KnowledgeWorks
lispworks-corba-5.1-*n*.i386.rpm	LispWorks ORB
lispworks-sql-5.1-*n*.i386.rpm	Common SQL

Table 4.3 File distributions for layered products in the Enterprise Edition

Install these modules as described in Section 4.4.3.4.

4.4.3.6 Documentation and saving space

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

Documentation is installed by default in the **lib/5-1-0-0/manual** sub-directory of the LispWorks installation directory.

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

```
rpm --install --excludedocs --prefix <directory>
lispworks-5.1-n.i386.rpm
```

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

```
rpm --install --prefix <directory> --replacepkgs
lispworks-5.1-n.i386.rpm
```

4.4.3.7 Installing Patches

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

4.4.4 Installation from the tar files

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

Table 4.4 Files for 32-bit Professional and Enterprise Editions

lw51-x86-linux.tar.gz	32-bit LispWorks image, modules and examples

lwdoc51-x86-linux.tar.gz	Documentation in HTML and PDF formats

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

To install from these files:

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

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

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

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

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

See Section 4.6 for how to enter your license details.

4.4.4.1 Installing Patches

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

4.5 LispWorks looks for a license key

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

For instructions on entering your license key, see Section 4.6.1, "Entering the license data" below.

For more information about license keys, see Section 9.2, "License keys".

4.6 Running LispWorks

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

The LispWorks executable is named as shown here:.

`lispworks-personal-5-1-0-x86-linux`	Personal Edition
`lispworks-5-1-0-x86-linux`	32-bit Professional or Enterprise Edition
`lispworks-5-1-0-amd64-linux`	64-bit Enterprise Edition

When you run LispWorks, the Lisp Monitor and splashscreen should appear, followed by the LispWorks Podium and a Listener. See "Troubleshooting" on page 87 for details if this does not happen.

4.6.1 Entering the license data

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

`lispworks-5-1-0-x86-linux --lwlicenseserial` *SERIALNUMBER* **`--lwlicensekey`** *LICENSEKEY*

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

```
LispWorks license installed successfully.
```

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

If you have received LispWorks on a CD-ROM then your license key is supplied on a label inside the CD-ROM case. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

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

4.7 Configuring the image

If you installed the Professional or Enterprise Edition of LispWorks, you can now configure your LispWorks image to suit your needs and load the Professional or Enterprise Edition modules as necessary. For instructions, see Chapter 9, "Configuration Details on Linux and FreeBSD".

4.8 Printable LispWorks documentation

In a default Professional/Enterprise installation, the **lib/5-1-0-0/manual/offline** directory contains PDF format versions of the manuals.

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

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

4.9 Uninstalling LispWorks for Linux

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

```
rpm --erase lispworks-5.1
```

If patches have been added via RPM, then you will first need to uninstall that package, which will be named **lispworks-patches5.1**. The same applies to additional RPM packages such as **lispworks-corba**.

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

If you installed LispWorks from the tar archives, simply do

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

4.10 Upgrading to LispWorks Enterprise Edition

You can upgrade from LispWorks Professional Edition by **Help > Register...** and enter an Enterprise license key.

5

Installation on FreeBSD

This chapter is an installation guide for LispWorks 5.1 for FreeBSD. Chapter 9, discusses post-installation and configuration in detail, but this chapter presents the instructions necessary to get LispWorks up and running on your system.

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
32MB of memory, preferably 64MB	FreeBSD 5.4 or later, or FreeBSD 6.0 or later with compat5x
160MB of disk space for Enterprise Edition plus documentation	OpenMotif 2.2.x

Table 5.1

Hardware Requirements	Software Requirements
	Netscape, Mozilla, FireFox or Opera Web browser for viewing on-line documentation

Table 5.1

5.1.1 Motif libraries

LispWorks 5.1 for FreeBSD requires that the X11 release 6 (or higher) OpenMotif (version 2.2 or higher) are installed on your machine. Install Open Motif 2.2.x from the FreeBSD ports tree. Your systems administrator may be able to help if you do not know how to do this.

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

5.1.2 Disk requirements during installation

LispWorks requires about 50MB to install without documentation. Installing the documentation adds about 110MB to this. A full installation of the Enterprise Edition with all documentation requires about 160MB.

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 License agreement

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

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

- For the Professional and Enterprise Editions, run

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

- For the Personal Edition, run:

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

Enter "yes" if you agree to the license terms.

5.3 Software on the CD-ROM

LispWorks 5.1 for FreeBSD is supplied as a standard package file in the **`x86-freebsd`** directory on your CD-ROM.

5.3.1 Professional and Enterprise Edition distributions

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

5.4 Installing LispWorks for FreeBSD

5.4.1 Main installation and patches

LispWorks Professional and Enterprise Editions are supplied as a standard software package file containing version 5.1. There may be a downloadable patch bundle which upgrades LispWorks to version 5.1.x. You need to complete the main installation before adding patches. The installer for 32-bit LispWorks contains both Professional and Enterprise Editions.

5.4.2 Information for Beta testers

Users of LispWorks 5.1 Beta should completely uninstall it (including any patches added to the initial beta installation) before installing LispWorks 5.1.

See "Uninstalling LispWorks for FreeBSD" on page 37 for instructions.

5.4.3 Installation software package file

If you already have LispWorks 5.1 Beta installed, please uninstall it before installing this product. See Section 5.9, "Uninstalling LispWorks for FreeBSD".

5.4.3.1 Installation directories

By default LispWorks is installed in **/usr/local/lib/LispWorks** and a symbolic link to the executable is placed in **/usr/local/bin/lispworks-5-1-0-x86-freebsd**. However, the software package is relocatable, and the **-p** option can be used to allow the installation of LispWorks in a user-specified directory. The default prefix is **/usr/local**.

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

5.4.3.2 Selecting the correct software package file

The LispWorks Professional/Enterprise software package file is called

```
lispworks-5.1.tgz
```

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

The Personal Edition software package file is called

```
lispworks-personal-5.1.tgz
```

5.4.3.3 Installing LispWorks for FreeBSD

To install LispWorks, perform the following steps as root:

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

   ```
   pkg_add lispworks-5.1.tgz
   ```

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

   ```
   pkg_add -p <directory> lispworks-5.1.tgz
   ```

   installs LispWorks in **<***directory***>**.

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

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

See Section 5.6 for instructions on entering your license details.

5.4.3.4 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-5.1.tgz
```

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

5.4.3.5 Installing Patches

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

5.5 LispWorks looks for a license key

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

For instructions on entering your license key, see Section 5.6.1, “Entering the license data” below.

For more information about license keys, see Section 9.2, “License keys”.

5.6 Running LispWorks

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

The LispWorks executable is named as shown here:.

lispworks-personal-5-1-0-x86-freebsd	Personal Edition
lispworks-5-1-0-x86-freebsd	Professional or Enterprise Edition

When you run LispWorks, the Lisp Monitor and splashscreen should appear, followed by the LispWorks Podium and a Listener. See "Troubleshooting" on page 87 for details if this does not happen.

5.6.1 Entering the license data

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

lispworks-5-1-0-x86-freebsd --lwlicenseserial *SERIALNUMBER* **--lwlicensekey** *LICENSEKEY*

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

```
LispWorks license installed successfully.
```

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

If you have received LispWorks on a CD-ROM then your license key is supplied on a label inside the CD-ROM case. If you have downloaded the product then your license key will be supplied to you in email from Lisp Support.

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

5.7 Configuring the image

If you installed the Professional or Enterprise Edition of LispWorks, you can now configure your LispWorks image to suit your needs and load the Professional or Enterprise Edition modules as necessary. For instructions, see Chapter 9, "Configuration Details on Linux and FreeBSD".

5.8 Printable LispWorks documentation

In a default Professional/Enterprise installation, the **`lib/5-1-0-0/manual/offline`** directory contains PDF format versions of the manuals.

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

5.9 Uninstalling LispWorks for FreeBSD

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

```
pkg_delete lispworks-5.1
```

If patches have been installed, then you will first need to uninstall that package, which will be named **`lispworks-patches5.1`**.

5.10 Upgrading to LispWorks Enterprise Edition

You can upgrade from LispWorks Professional Edition by **Help > Register...** and enter an Enterprise license key.

6

Installation on UNIX

6.1 Introduction

This chapter is a brief installation guide for UNIX LispWorks 5.1. Chapter 10 discusses installation and configuration in detail, but this chapter presents the minmum instructions necessary to get LispWorks up and running on your system. If you have difficulties installing LispWorks from these instructions, refer to the main guide, starting at Chapter 10, "Configuration Details on UNIX".

6.2 Extracting software from the CD-ROM

UNIX LispWorks 5.1 is supplied on a CD-ROM with the associated products CLIM 2.0, KnowledgeWorks, and LispWorks ORB. You will need root access while installing these products.

6.2.1 Finding out which CD-ROM files you need

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

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

Table 6.1 Platforms and associated codes

For HP PA (HP-UX 11x) you need the files named **lw51-hp-pa.tar** and **lwdoc51-unix.tar**.

For Sun Sparc (32-bit) you need the files named **lw51-sparc.tar** and **lwdoc51-unix.tar**.

For Sun Sparc (64-bit) you need the files named **lw51-sparc64.tar** and **lwdoc51-sparc64.tar**.

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

6.2.2 Unpacking the CD-ROM files

To unpack the CD-ROM files:

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

```
% tar -xof filename
```

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

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

Thus, an image named **lispworks-5-1-0-hp-pa11** would be a LispWorks 5.1 image for use on an HP PA machine running HP-UX 11.

6.3 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/5-1-0-0** alongside the image.

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

1. Put the LispWorks distribution in **/usr/lib/lispworks**. You will then have the LispWorks image at top-level in the **/usr/lib/lispworks** directory, and subdirectories **/usr/lib/lispworks/lib/5-1-0-0**.

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

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

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

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

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

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

See Section 6.5 below for more information about configuring LispWorks. You will need to install your license key first.

6.4 Obtaining and Installing your license keys

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

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

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

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

6.4.1.1 If you are using the keyfile system

You will need a valid key, placed in a keyfile, for LispWorks to run. Note that keys and licenses issued for use with LispWorks version 4.x do not work for LispWorks 5.1.

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

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

lisp-keys@lispworks.com

Other queries should be sent to

lisp-support@lispworks.com

although please be sure to check Section 11.8, “Reporting bugs” for instructions before sending us a bug report. If you do not have e-mail access, you can contact Lisp Support by telephone or ordinary postal mail. Contact details are in Section 11.8.8, “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/5-1-0-0/config/keyfile.***hostname*, where ***hostname*** is the name of the host machine on which LispWorks is to run. The **lib** directory is expected by default to be located at **/usr/lib/lispworks/lib** (see Section 6.3 above)
- **lib/5-1-0-0/config/keyfile**, where the **lib** directory is as above.

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

For more details, see "How to obtain keys" on page 79.

6.4.1.2 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*.

6.5 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.

6.5.1 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-5-1-0-sparc-solaris -siteinit - -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 42

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.

6.5.2 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"**.

6.6 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 40.

HTML documentation is installed in the **lib/5-1-0-0/manual/online** subdirectory of the LispWorks library, and can be accessed via the **Help** menu in the Common LispWorks IDE.

The PDF format manuals are installed in the **lib/5-1-0-0/manual/offline/pdf** subdirectory of the LispWorks library.

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

To use each of Delivery, LispWorks ORB, CLIM 2.0 and KnowledgeWorks you must obtain the required key and put in your keyfile. See “Keyfiles and the license server for HP PA and Sun Sparc (32-bit)” on page 42.

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 44.

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

7

Configuration Details on Mac OS X

7.1 Introduction

This chapter explains how to get your LispWorks Professional or Enterprise Edition up and running, having already installed the files from the CD-ROM into an appropriate folder. If you have not done this, refer to Chapter 2, "Installation on Mac OS X".

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

This chapter covers the following topics:

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

7.2 License keys

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

The image looks for a valid license key in the following places, in order:

- in the current working directory (folder)
- in the directory containing the LispWorks executable
- in the **`Library/lib/5-1-0-0/config`** subdirectory of the LispWorks installation directory

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

7.3 Configuring your LispWorks installation

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

7.3.1 Levels of configuration

There are two levels of configuration:

- configuring and resaving the image, thereby creating a new image that is exactly as you want it at startup
- 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.

7.3.2 Configuring images for the different GUIs

If you have installed both the LispWorks images, for native Mac OS X and for X11/Motif, you will want to configure two images.

If necessary your Lisp configuration and initialization files can run code for one image or the other by conditionalization on the feature **:cocoa**. The native Mac OS X LispWorks image has **:cocoa** on ***features*** while the X11/Motif LispWorks image does not.

7.3.3 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 7.4, below, and Section 7.5, "Initializing LispWorks" for further details.

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

config/a-dot-lispworks.lisp is a sample personal initialization file. You might like to copy this 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 7.4, below, and Section 7.5, "Initializing LispWorks" for further details.

7.4 Saving and testing the configured image

Make a copy of **config/configure.lisp** called **/tmp/my-configuration.lisp**. When you have made the desired changes in **my-configuration.lisp** you can save a new LispWorks image. To do this, follow the instructions below.

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

```
(load-all-patches)
(load "/tmp/my-configuration.lisp")
#+:cocoa
(compile-file-if-needed
 (sys:example-file "configuration/macos-application-bundle")
 :load t)
(save-image #+:cocoa
            (write-macos-application-bundle
             "/Applications/LispWorks 5.1/My LispWorks.app")
            #-:cocoa
            "my-lispworks-motif")
```

   Note 1: The use of example code supplied with LispWorks which creates a Mac OS X application bundle. This code is in the example file **examples/configuration/macos-application-bundle.lisp**

Note 2: This will create a non-universal binary, containing only the architecture on which you call **save-image**.

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

   ```
   % cd "/Applications/LispWorks 5.1/LispWorks.app/Contents/MacOS"
   ```

   or for the X11/Motif LispWorks image:

   ```
   % cd "/Applications/LispWorks 5.1"
   ```

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

   ```
   % lispworks-5-1-0-macos-universal -siteinit - -build
   /tmp/save-config.lisp
   ```

   or

   ```
   % lispworks-5-1-0-macos-universal-motif -siteinit - -build
   /tmp/save-config.lisp
   ```

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

   Note that the command line also suppresses the **siteinit** because this will be loaded automatically when you start the configured image.

   Saving the image takes some time.

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

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

7.4.1 Testing the newly saved image

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

1. If you are using an X11/Motif image, change directory to **/tmp**.

2. When using X11, verify that your **DISPLAY** environment variable is correctly set and that your machine has permission to connect to the display.
3. Start up the new image, by entering the path of the X11/Motif executable or by double-clicking on the LispWorks icon in the Mac OS X Finder.

   The window-based environment should now initialize—during initialization a window displaying a copyright notice will appear on the screen.

   You may wish to work through some of the examples in the *LispWorks User Guide*, to further check that the configured image has been successfully built.
4. Test the load-on-demand system. In the Listener, type:

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

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

7.4.2 Saving a non-windowing image

For some purposes such as scripting it is convenient to have a LispWorks image that does not start the graphical programming environment.

To save an image which does not automatically start the GUI, use a script as described in "Saving and testing the configured image" on page 50 but pass the **:environment** argument to **save-image**. For example:

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

7.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. 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 5.1/LispWorks.app/Contents/MacOS/lispw
orks-5-1-0-macos-universal" -init my-lisp-init
```

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

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

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

```
% "/Applications/LispWorks 5.1/LispWorks.app/Contents/MacOS/lispw
orks-5-1-0-macos-universal" -init - -siteinit -
```

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

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

7.6 Loading CLIM 2.0

Load CLIM 2.0 into a LispWorks for X11/Motif image with

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

and the CLIM demos with

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

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

```
(load-all-patches)
(require "clim")
(save-image "<destination>/clim-lispworks")
```

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

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

Note: CLIM is not supported by the LispWorks native Mac OS X image and cannot be loaded into it.

Note: Do not attempt to load CLIM via the clim loader files in the clim distribution. This will cause CLIM patches to not be loaded. Use **`(require "clim")`**.

7.7 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*.

7.7.1 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* for further information.

7.7.2 Supported databases

Common SQL on Mac OS X has been tested with DBMS Postgres 7.2.1, MySQL 5.0.18, Oracle Instant Client 10.1.0.3, ODBC driver PSQLODBC development code, and IODBC as supplied with Mac OS X.

7.7.3 Special considerations when using Common SQL

7.7.3.1 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.

7.7.3.2 Errors using PSQLODBC

The PSQLODBC driver, when it does not find any of the Servername, Database or Username in `.odbc.ini`, returns the wrong error code. This tells the calling function that the user cancelled the login dialog.

Therefore, if Common SQL reports that the user cancelled when trying to connect, you need to check that you have got Servername, Database and Username, with the correct case, in the section for the datasource in the `.odbc.ini` file.

Note: Username may alternatively be given in the connect string.

7.7.3.3 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.

7.7.3.4 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).

7.8 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.

8

Configuration Details on Windows

8.1 Introduction

This chapter explains how to get your LispWorks Professional or Enterprise Edition up and running, having already installed the files from the CD-ROM into an appropriate directory. If you have not done this, refer to Chapter 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”

8.2 License keys

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

8.3 Configuring your LispWorks installation

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

8.3.1 Levels of configuration

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

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

In the first case, you use edited copies of files in the **`config`** folder to achieve your aims.

In the second case, you make entries in your initialization file. This is a file read every time LispWorks starts up, and it can contain any valid Common Lisp code. (Most of the configurable settings in LispWorks can be controlled from Common Lisp.) Your initialization file can be changed via **`Tools > Global Preferences...`** in the Common LispWorks IDE.

8.3.2 Configuration files available

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

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

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

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

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

On startup, the image loads **siteinit.lisp** and your initialization file, in that order. The command line options **-siteinit** and **-init** can be used to specify loading of different files or to suppress them altogether. See the example in Section 8.4, below, and Section 8.5, "Initializing LispWorks" for further details.

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

config/a-dot-lispworks.lisp is a sample personal initialization file. You might like to copy this 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 8.4, below, and Section 8.5, "Initializing LispWorks" for further details.

8.4 Saving and testing the configured image

Make a copy of **config\configure.lisp** called **C:\temp\my-configuration.lisp**. When you have made any desired changes in **my-configura-**

tion.lisp you can save a new LispWorks image. To do this, follow the instructions below.

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

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

2. Change directory to the LispWorks installation directory, for example:

```
C:

cd C:\Program Files\LispWorks
```

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

```
C:\Program Files\LispWorks>lispworks-5-1-0-x86-win32.exe
-siteinit - -build C:\temp\save-config.lisp
```

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

Note that the command line also suppresses the **siteinit** because this will be loaded automatically when you start the configured image.

Saving the image takes some time.

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

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

8.4.1 Testing the newly saved image

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

1. Start up the new image.

The window-based environment should now initialize—during initialization a window displaying a copyright notice will appear on the screen.

You may wish to work through some of the examples in the *LispWorks User Guide*, to further check that the configured image has been successfully built.

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

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

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

8.4.2 Saving a non-windowing image

For some purposes such as scripting it is convenient to have a LispWorks image that does not start the graphical programming environment.

To save an image which does not automatically start the GUI, use a script as described in "Saving and testing the configured image" on page 59 but pass the **:environment** argument to **save-image**. For example:

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

8.5 Initializing LispWorks

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

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

```
C:\Program Files\LispWorks>lispworks-5-1-0-x86-win32.exe -init
my-lisp-init
```

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

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

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

```
C:\Program Files\LispWorks>lispworks-5-1-0-x86-win32.exe -init -
-siteinit -
```

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

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

8.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 5.1 with

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

and the CLIM demos with

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

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

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

```
(load-all-patches)
(require "clim")
(save-image "<destination>/clim-lispworks")
```

8.6.1 Running the CLIM demos

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

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

This 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*

8.7 The Common SQL interface

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

8.7.1 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* for further information.

8.8 Common Prolog and KnowledgeWorks

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

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

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

9

Configuration Details on Linux and FreeBSD

9.1 Introduction

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

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

This chapter covers the following topics:

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

- "Common Prolog and KnowledgeWorks"

9.2 License keys

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

The image looks for a valid license key in the following places, in order:

- in the current working directory
- in the directory containing the LispWorks executable
- in the **`lib/5-1-0-0/config`** subdirectory of the LispWorks installation directory

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

9.3 Configuring your LispWorks installation

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

9.3.1 Levels of configuration

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

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

In the first case, you use edited copies of files in the **`config`** directory to achieve your aims.

In the second case, you make entries in your initialization file. This is a file read every time LispWorks starts up, and it can contain any valid Common Lisp code. (Most of the configurable settings in LispWorks can be controlled from Common Lisp.) By default the file is called **.lispworks** and is in your home directory. Your initialization file can be changed via **Tools > Global Preferences...** in the Common LispWorks IDE.

9.3.2 Configuration files available

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

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

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

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

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

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

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

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

9.4 Saving and testing the 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. To do this, follow the instructions below.

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-5-1-0-x86-linux -siteinit - -build
   /tmp/save-config.lisp
   ```

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

Note that the command line also suppresses the **siteinit** because this will be loaded automatically when you start the configured image.

Saving the image takes some time.

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

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

9.4.1 Testing the newly saved image

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

1. Change directory to **/tmp**.
2. Verify that your **DISPLAY** environment variable is correctly set and that your machine has permission to connect to the display.
3. Start up the new image.

   The window-based environment should now initialize—during initialization a window displaying a copyright notice will appear on the screen.

   You may wish to work through some of the examples in the *LispWorks User Guide*, to further check that the configured image has been successfully built.
4. Test the **load-on-demand** system. In the Listener, type:

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

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

9.4.2 Saving a non-windowing image

For some purposes such as scripting it is convenient to have a LispWorks image that does not start the graphical programming environment.

To save an image which does not automatically start the GUI, use a script as described in "Saving and testing the configured image" on page 68 but pass the **:environment** argument to **save-image**. For example:

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

9.5 Initializing LispWorks

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is **~/.lispworks** by default. ~ 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-5-1-0-x86-linux -init my-lisp-init
```

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

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

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

```
% lispworks-5-1-0-x86-linux -init - -siteinit -
```

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

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

9.6 Loading CLIM 2.0

Load CLIM 2.0 into LispWorks 5.1 with

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

and the CLIM demos with

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

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

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

```
(load-all-patches)
(require "clim")
(save-image "<destination>/clim-lispworks")
```

9.6.1 Running the CLIM demos

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

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

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

9.7 The Common SQL interface

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

9.7.1 Loading the Common SQL interface

To load the Common SQL interface to use ODBC enter:

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

and at runtime call:

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

and then you can connect to any installed ODBC datasource.

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

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

and at runtime call:

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

See the *LispWorks User Guide* for further information.

9.8 Common Prolog and KnowledgeWorks

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

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

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

9.9 Documentation for LispWorks for FreeBSD

Except where explicitly mentioned, information stated as specific to LispWorks for Linux applies just the same to LispWorks for FreeBSD.

10

Configuration Details on UNIX

10.1 Disk requirements

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

Installing the documentation adds up to 66MB to this. You can delete some of these files if you wish, for example you might not need the PDF manuals in **lib/5-1-0-0/manual/offline/pdf** (28Mb). You can download these PDF format manuals from **www.lispworks.com/documentation** at any time, and the same manuals are also available there in PostScript format. However, note that the **Help** menu commands will not work if you corrupt the **lib/5-1-0-0/manual/online** directory of the LispWorks library.

10.2 Software Requirements

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

10.3 The CD-ROM

This section explains the organization of the LispWorks 5.1 CD-ROM which contains the LispWorks products you have bought, and how to mount it.

10.3.1 The LispWorks 5.1 CD-ROM

The CD-ROM contains images for LispWorks 5.1 and associated products on your platform or platforms.

10.3.1.1 CD-ROM format

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

10.3.2 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 74.
2. Extract the product files from the **`tar`** file containing them. This is described in "Installing LispWorks" on page 75.

10.3.3 Mounting the CD-ROM

Before you can access the files on the CD-ROM, it has to be mounted onto your UNIX filesystem. You may need root access on your machine to do this.

On some platforms, the CD-ROM will be mounted automatically when you place it in the drive. On most, however, you will have to run a mounting program to mount it. You may also have to create a directory on your machine to serve as the mount point. (The mount point is the point in your filesystem at which the CD-ROM directory structure will be found.)

When you have mounted the CD-ROM and can see the **`tar`** files on your UNIX filesystem, you are ready to unpack them. Once you are finished with the **`tar`** files on the CD-ROM, you can remove it from your drive, but only after you have performed an "unmount" operation.

When unmounting it is necessary that no process has the CD-ROM mount point as the current directory, and again, root access is necessary. Pushing the eject button on the drive may not do anything until the volume has been unmounted.

The basic syntax of the mounting and unmounting operations on each supported platform is given in each of the platform-specific sections below.

10.3.3.1 HP UX (HP Precision Architecture)

To mount:

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

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

To unmount:

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

Again, use the appropriate device designation for your hardware.

10.3.3.2 Solaris (Sun Sparc)

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

```
/cdrom/lw_51/
```

To unmount:

```
umount /cdrom/lw_51/
```

10.4 Installing LispWorks

This section explains how to install LispWorks, having already mounted the CD-ROM. If you have not done this, refer to Section 10.3, "The CD-ROM". It also describes how you obtain keys to run LispWorks 5.1.

10.4.1 Unpacking the TAR files

Once the CD-ROM is mounted, you can begin to unpack the **tar** files for the products you have purchased. You will need root access to do this.

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

10.4.1.1 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.

10.4.1.2 Keeping your old LispWorks installation

You can install LispWorks 5.1 in the same directory as previous versions of LispWorks such as LispWorks 5.0. This is because all the 5.1 files are stored in a subdirectory called **lib/5-1-0-0**.

You must recompile all your code with the LispWorks 5.1 compiler.

Binaries produced by **compile-file** in previous versions of LispWorks such as LispWorks 5.0 do not load into a LispWorks 5.1 image.

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

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

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

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

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

For example, to extract the files for LispWorks (32-bit) on Solaris, with the CD-ROM mounted at **/cdrom/lw_51/**, you would type

```
tar -xof /cdrom/lw_51/lw51-sparc.tar
```

10.4.1.4 HP UX (HP Precision Architecture)

The files you need to unpack for LispWorks on HP UX are **lw51-hp-pa.tar** and **lwdoc51-unix.tar**.

The LispWorks image is

```
./lispworks-5-1-0-hp-pa11
```

10.4.1.5 Solaris (LispWorks 32-bit)

The files you need to unpack for LispWorks (32-bit) on Solaris are **lw51-sparc.tar** and **lwdoc51-unix.tar**.

The LispWorks image is

```
./lispworks-5-1-0-sparc-solaris
```

10.4.1.6 Solaris (LispWorks 64-bit)

The files you need to unpack for LispWorks (64-bit) on Solaris are **lw51-sparc64.tar** and **lwdoc51-sparc64.tar**.

The LispWorks image is

```
./lispworks-5-1-0-sparc64-solaris
```

10.4.2 Keyfiles and how to obtain them

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

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

10.4.2.1 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/5-1-0-0/config** (see "If you are using the keyfile system" on page 42.
- **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/5-1-0-0/config**.

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

10.4.2.2 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.

10.4.2.3 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 98.

Once you have the key, write it into a file in one of the places listed in Section 10.4.2.1, and start up the LispWorks image.

10.4.3 The License Server

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

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.

10.5 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.

10.5.1 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-5-1-0-sparc-solaris** is the LispWorks 5.1 image for use on Sun Sparc (32-bit) Solaris machines.

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

As noted in Section 10.4.1.1 on page 76, 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**.

10.5.2 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/5-1-0-0/**.

Among the directories on this subdirectory are the following:

- **config** — various files that can be adjusted in order to customize the image (see Section 10.7 on page 81).
- **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 PDF versions of the complete LispWorks manual set.

By default, all these directories are assumed to reside beneath **/usr/lib/lispworks/lib/5-1-0-0/**, although you may place the **lib** directory somewhere else.

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

10.6 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 PDF versions of each manual.

10.7 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.)

10.7.1 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.

10.7.2 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 10.7.3 below, and see also Section 10.8, "LispWorks initialization arguments" for further details.

10.7.3 Saving and testing the 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. To do this, follow the instructions below.

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-5-1-0-sparc-solaris -init - -siteinit -
   ```

   If the image will not run at this stage, it is probably not finding a valid key. See "Keyfiles and how to obtain them" on page 77.

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.

10.7.3.1 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* to check further that the configured image has successfully built.

10.8 LispWorks initialization arguments

When LispWorks starts up, it looks for an initialization file to load. The name of the file is held in ***init-file-name***, and is "**~/.lispworks**" by default. The file may contain any valid Lisp code.

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

```
unix% lispworks -init my-lisp-init
```

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

Alternatively, an initialization file may be specified by setting the UNIX environment variable **LW_INIT**. If set, the specified file will be used instead of that named by ***init-file-name***.

The loading of the siteinit file (located by default at **config/siteinit.lisp**) may similarly be controlled either by the **-siteinit** command line argument, or the **LW_SITE_INIT** variable and ***site-init-file-name***.

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

```
unix% lispworks -init - -siteinit -
```

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

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

11

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.

11.1 Troubleshooting

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

11.1.1 License key errors in the Professional and Enterprise Editions

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

These are the possible problems:

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

On Linux and FreeBSD, this is also a possible cause of the problem:

- The machine name has changed since LispWorks was installed.

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

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

11.1.2 Failure of the load-on-demand system

Module files are in the modules directory **lib/5-1-0-0/load-on-demand** under ***lispworks-directory***.

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

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

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

11.1.3 Memory requirements

To run the full LispWorks system, with its GUI, you will need around 20MB of swap space for the image and whatever else is necessary to accommodate your application.

When running a large image, you may occasionally see

```
<**> Failed to enlarge memory
```

printed to the standard output.

The message means that the LispWorks image attempted to expand one of the GC generations, but there was not enough swap space to accommodate the resulting growth in image size. When this happens, the garbage collector is invoked, and it will usually manage to free the required space.

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

Occasionally, however, continued demand for additional memory will end up exhausting resources. You will then see the message above repeatedly, and there will be little or no other activity apparent in the image. At this point you should restart the image, or increase swap space. In cases where external libraries are mapped above LispWorks and inhibit its growth, you may be able to relocate LispWorks, as described under "Startup relocation" in the *LispWorks User Guide*.

11.2 Troubleshooting on Mac OS X

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

If you're using the LispWorks image with the X11/Motif GUI, see also Section 11.6, "Troubleshooting on X11/Motif" below for issues specific to X11/Motif.

11.2.1 Default installation requires administrator on Mac OS X

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

However, a non-administrator may install LispWorks elsewhere.

11.2.2 Failure to start when disconnected from the Internet

By default MacOS X machines have different names when connected and when not connected. After changing the connection state, the 64-bit LispWorks license check will fail, because the data is encoded with the machine name.

The machine name is configured by the line

```
HOSTNAME=-AUTOMATIC-
```

in **/etc/hostconfig**.

The recommended fix is to edit **/etc/hostconfig** to give your machine a fixed hostname, then reset the license file if necessary by running in Terminal.app:

```
$ ./lispworks-5-1-0-macos64-universal --lwlicenseserial
SERIALNUMBER --lwlicensekey LICENSEKEY
```

where *SERIALNUMBER* and *LICENSEKEY* are the strings supplied with LispWorks. Then the LispWorks license check will always lookup the expected hostname.

Note: this section does not apply to 32-bit LispWorks for Macintosh.

11.2.3 Text displayed incorrectly in the editor on Mac OS X

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

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

Select the font in an Editor or Listener tool by **Window > Window Preferences... > Font**.

11.3 Troubleshooting on Linux

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

See also "Troubleshooting on X11/Motif" on page 94 below for issues specific to X11/Motif.

11.3.1 Processes hanging

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

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

LispWorks 5.1 supports kernel versions 2.4.20 and later.

11.3.2 RPM_INSTALL_PREFIX not set

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

```
# rpm --install tmp/lispworks-clim-5.1-1.i386.rpm
Environment variable RPM_INSTALL_PREFIX not set, setting it to
/usr
LispWorks installation not found in /usr.
error: %pre(lispworks-clim-5.1-1) scriptlet failed, exit status 1
error:   install: %pre scriptlet failed (2), skipping lispworks-
clim-5.1-1
#
```

This is only a problem when LispWorks itself was installed in a non-default location (that is, using the **--prefix** RPM option). You would then want to supply that same **--prefix** value when installing the secondary rpm. A bug in RPM means that a required environment variable **RPM_INSTALL_PREFIX** is not set automically to the supplied value. We have seen this bug in RPM version 4.2, as distributed with RedHat 8 and 9.

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

```
rpm --install --prefix /usr/lisp lispworks-5.1-1.i386.rpm
```

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

```
setenv RPM_INSTALL_PREFIX /usr/lisp
rpm --install --prefix /usr/lisp lispworks-clim-5.1-1.i386.rpm
```

11.3.3 Using multiple versions of Motif on Linux

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

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

You may wish to maintain multiple versions of the Motif/LessTif libraries in order to run various applications simultaneously. However, because the filenames of the libraries can conflict, this can only be done by installing libraries in non-standard locations.

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

Note: to find out which version of libXm your LispWorks 5.1 image is actually using, look in the bug form. See "Generate a bug report template" on page 99 for instructions on generating the bug form.

11.3.4 Using LispWorks for Linux on FreeBSD

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

LispWorks for FreeBSD is now available, however.

11.4 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 94 below for issues specific to X11/Motif.

11.4.1 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.

11.5 Troubleshooting on UNIX

This section describes some of the most common problems that can occur during installation or configuration of LispWorks for UNIX (not including Linux).

See also "Troubleshooting on X11/Motif" on page 94 for issues specific to X11/Motif.

11.5.1 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:

```
lwdoc51-unix.tar;1
```

then check the **mount** command used ("Mounting the CD-ROM" on page 74).

11.5.2 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/5-1-0-0**) or be executed in the same directory as the keyfile. If the image has been configured, check that the keyfile is in the right place and that the value of ***lispworks-directory*** is correct.

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

11.6 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.

11.6.1 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.

11.6.2 Problems with fonts

Running under X11, LispWorks may print a message saying that it is unable to open a font and is using a default instead. The environment will still run but it may not always use the right font.

LispWorks comes configured with the fonts most commonly found with the target machine type. However the fonts supplied vary between implementations and installations. The fonts available on a particular server can be determined by using the **`xlsfonts(1)`** command. Fonts are chosen based on the X11 resources. See “X11 resources” on page 95 for more information.

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

11.6.3 Problems with colors

Running under X11, on starting up the environment, or any tool within it, LispWorks may print a message saying that a particular color could not be allocated.

This problem can occur if your X color map is full. If this is the case, LispWorks cannot allocate all the colors that are specified in the X11 resources.

This may happen if you have many different colors on your screen, for instance when displaying a picture in the root window of your display.

Colors are chosen based on the X11 resources. See "X11 resources" on page 95 for more information.

To remove the problem, you can then change the resources (for example, by editing the file mentioned in "X11 resources" on page 95) to reduce the number of colors LispWorks allocates.

11.6.4 Mnemonics and Alt

Motif hardwires its mnemonic processing to use **mod1**, so we disable mnemonics if that is Lisp's **Meta** modifier to allow the Emacs-style editor to work. (The accelerator code uses the same keyboard mapping check as the mnemonics so **Alt** accelerators would also get disabled if you had them.)

11.6.5 Non-standard X11 key bindings

When using X11/Motif, if you want Emacs-style keys **Ctrl-n, Ctrl-p** in LispWorks list panels, such as the Editor's buffers view, add the following to the X11 resources (see Section 11.6.6):

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

11.6.6 X11 resources

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

11.6.7 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 5.1/lispworks-5-1-0-macos-
universal-motif can't open library:
/usr/local/lib/libXm.4.dylib (No such file or directory, errno
= 2)
.
```

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

11.7 Updating with patches

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

11.7.1 Extracting simple patches

Save the email attachment to your disk.

See Section 11.7.3.2, "Private patches" below about location of your private patches.

11.7.2 If you cannot receive electronic mail

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

11.7.3 Different types of patch

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

11.7.3.1 Public patches

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

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

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

You should consider saving a new image with the latest patches pre-loaded, as described in Section 7.4, “Saving and testing the configured image” (Mac OS X), Section 8.4, “Saving and testing the configured image” (Windows) or Section 9.4, “Saving and testing the configured image” (Linux), or Section 10.7.3, “Saving and testing the configured image” (non-Linux UNIX).

11.7.3.2 Private patches

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

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

<*install*>/LispWorks 5.1/Library/lib/5-1-0-0/private-patches/my-patch.nfasl

On Windows:

<*install*>\lib\5-1-0-0\private-patches\my-patch.ofasl

On Linux:

<*install*>/lib/5-1-0-0/private-patches/my-patch.ufasl

On UNIX:

<*install*>/lib/5-1-0-0/private-patches/my-patch.pfasl (for HP-PA)
<*install*>/lib/5-1-0-0/private-patches/my-patch.wfasl (for SPARC)

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

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

This form should be added in the file:

private-patches/load.lisp

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

You may choose to save a reconfigured image with the new patch loaded - for details see the instructions in Section 7.4, “Saving and testing the configured image” (Mac OS X), Section 8.4, “Saving and testing the configured image” (Windows), Section 9.4, “Saving and testing the configured image” (Linux), or Section 10.7.3, “Saving and testing the configured image” (non-Linux UNIX). You can alternatively choose to load the patch file on startup. The option you choose will depend on how many people at your site will need access to the new patch, and how many will need access to an image without the patch loaded.

11.8 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 11.8.8. Please note that we much prefer email.

11.8.1 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**.

11.8.2 Performance Issues

If the problem is poor performance, you should use **room**, **extended-time** and **profile** to check what actually happens. See the *LispWorks 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.

11.8.3 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.

11.8.4 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 Common LispWorks IDE, and you are receiving too many notifiers, set **`dbg::*full-windowing-debugging*`** to **`nil`** and try again. This will cause the console version of debugger to be used instead.
- If the problem occurs when compiling or loading a large system, call (**`toggle-source-debugging nil`**) and try again.
- If you ever receive any unexpected terminal output starting with the characters **`<**>`**, please send all of the output—however much there is of it.

  Note: terminal output is that written to **`*terminal-io*`**. Normally this is not visible when running the Mac OS X native GUI or the Windows GUI, though it is displayed in a Terminal.app or MS-DOS window if necessary.

11.8.5 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 5.1/LispWorks.app/Contents/MacOS/lispw
orks-5-1-0-macos-universal" -init init.lisp > lw.out
```

where `%` denotes a Unix shell prompt.

On Windows:

```
C:\> "Program Files\LispWorks\lispworks-5-1-0-x86-
win32.exe" -init init.lisp > lw.out
```

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

On Linux:

```
% /usr/bin/lispworks-5-1-0-x86-linux -init init.lisp > lw.out
```

where `%` denotes a Unix shell prompt.

On UNIX (SPARC in this example):

```
% /usr/lib/lispworks/lib/5-1-0-0/config/lispworks-5-1-0-sparc-
solaris -init init.lisp > lw.out
```

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

11.8.6 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.

11.8.7 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.)

11.8.8 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 99 for details. You can always get a backtrace from within the debugger by entering `:bb` at the debugger prompt

11.8.9 Sending large files

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

11.8.10 Information for Personal Edition users

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

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

12

Release Notes

12.1 Additional support for 64-bit on Macintosh

12.1.1 64-bit Cocoa GUI

LispWorks 5.1 (64-bit) for Macintosh runs only on Mac OS X 10.5, and supports a Cocoa-based GUI. The alternative X11/Motif GUI is still available.

Choose at install time to install just one, or both, of these GUI options.

12.1.2 Universal binaries on Macintosh

The supplied LispWorks (64-bit) for Macintosh images are now universal binaries which run the correct native architecture on PowerPC and Intel-based Macintosh computers by default. This was already true for LispWorks 5.0 (32-bit) for Macintosh but was not possible at the time of the initial LispWorks 5.0 (64-bit) for Macintosh release.

A running Lisp image only supports one architecture, chosen when the image was started. On a PowerPC based Macintosh, this is always the PowerPC architecture. On an Intel-based Macintosh, for 32-bit LispWorks it can be either the native Intel architecture or the PowerPC architecture (using Rosetta), while for 64-bit LispWorks it is always the Intel architecture.

Functions such as **save-image** and **deliver** create an image containing only the running architecture and functions that operate on fasl files such as **compile-file** and **load** only support the running architecture.

12.2 Running on 64-bit machines

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

12.3 New CAPI features

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

12.3.1 OLE embedding

LispWorks for Windows CAPI applications can now operate as ActiveX controls. See **capi:define-ole-control-component** and **capi:ole-control-component**.

In LispWorks 5.0 and previous versions, it is possible to embed an ActiveX control in a CAPI window, but not vice versa.

12.3.2 Drag and drop

CAPI now supports drag and drop for text and files on Windows and Cocoa.

Initiate dragging by calling **capi:drag-pane-object**. Provide a **:drop-callback** (as documented under **capi:simple-pane**) and use **capi:drop-...** functions to control dropping behavior.

12.3.3 "metafile" support on Cocoa

LispWorks 5.1 for Macintosh supports the metafile APIs **capi:draw-metafile**, **capi:with-external-metafile** and so on.

There are some restrictions on the *bounds* parameter - see the *LispWorks CAPI Reference Manual* for details.

12.3.4 Keyboard shortcuts on Cocoa

By default the standard shortcut **Command+Shift+P** now invokes a **Page Setup...** menu command in CAPI interfaces.

Return now invokes the *action-callack* of a **capi:list-panel** and of a **capi:tree-view**.

Typing now searches for an item in a **capi:list-panel**.

12.3.5 The Mac OS X Dock menu

You can control your LispWorks for Macintosh application's Dock menu using the new **:dock-menu** initarg of **capi:cocoa-default-application-interface**.

12.3.6 The Application Services menu

The **LispWorks > Services** menu in LispWorks for Macintosh now has support for standard Cocoa controls (but not **capi:output-pane** or **capi:editor-pane**).

12.3.7 Vista icons

capi:load-icon-image now supports 256x256 icons on Windows Vista. PNG compressed icons are supported.

12.3.8 New class supports filtering

The new class **capi:filtering-layout** adds a text pane allowing the user to filter the displayed data (such as items of a list) and to control how the filter operates.

12.3.9 Color in list items on Windows

capi:list-panel now supports color in the list items on Microsoft Windows. To use this you supply a suitable value for the new **:color-function** initarg.

12.3.10 Automatic resizing of pinboard objects

You can now specify that a pinboard object should resize itself automatically when its pinboard layout is resized. Set the various resize parameters for each object using **capi:set-object-automatic-resize**.

12.3.11 Better control over size of tab and switchable layouts

capi:tab-layout now allows you to specify that the initial size depends on the constraints of all the tabs, which can prevent unexpected resizing later when switching tabs.

Similarly **capi:switchable-layout** allows you to specify that the initial size depends on the constraints of the child panes.

For details see the new initarg **:combine-child-constraints**.

12.3.12 Browsing for application bundles

On Cocoa **capi:prompt-for-file** now supports selection of application bundle (**.app** folders) as files, if they match the filter.

12.3.13 editor-pane change callback

capi:editor-pane now supports a change callback via its new **:change-callback** initarg. See the example in **lib/5-1-0-0/examples/capi/editor/change-callback.lisp**.

12.3.14 editor-pane input callbacks

capi:editor-pane now supports input callbacks via new initargs **:before-input-callback** and **:after-input-callback**.

These callbacks are called when **capi:call-editor** is called.

12.3.15 New ways to control line wrapping in editor-pane

capi:editor-pane now supports three new initargs controlling line wrapping of the text it displays:

:line-wrap-marker specifies the marker to display at the end of a line that is wrapped.

:line-wrap-face specifies the text style to use to display the marker.

:wrap-style allows you to control whether long lines are wrapped normally (that is, splitting words), or wrapped at spaces so that whole words are displayed, or simply truncated.

Compatibility Note: The **:wrap-style** initarg supersedes **editor:set-window-split-on-space**, which is deprecated.

12.3.16 Help button for text input panes

capi:text-input-pane now supports a help toolbar button, with control over how help is presented to the user.

12.3.17 Finding the active interface

The new function **capi:screen-active-interface** returns the currently active interface on a given screen or (on Windows) a MDI document container.

12.3.18 Controlling window titles

New functions allow more control over the titles of CAPI interface windows. For an overview see the section "Window titles" in the *LispWorks CAPI User Guide*.

12.3.19 Avoiding updates on a destroyed interface

The new function **capi:execute-with-interface-if-alive** is useful for automatic updating of interfaces that may be destroyed by the user, where the update is redundant if the interface is not alive.

12.3.20 OpenGL example updated

The OpenGL/CAPI example now supports OpenGL version 2.1. Also the code in **loader.lisp** has been simplified for Linux and now also works on all LispWorks platforms, including FreeBSD, Solaris and HP-UX.

The example code is in **lib/5-1-0-0/examples/opengl**. Follow the instructions in **doc.txt** to define **capi:opengl-pane**.

You may wish to add more FLI definitions for use by your application.

12.4 Other CAPI changes

12.4.1 Passing initargs to interface components

Passing initargs down to the panes, layouts and menus of a **capi:interface** via initargs to the interface itself is now fully documented. See **capi:define-interface** in the *LispWorks CAPI Reference Manual*.

12.4.2 Reselection of single selection choice items

Choices such as **capi:list-panel** and **capi:option-pane** with *interaction* **:single-selection** now always call their selection callback when the user selects the selected item, regardless of whether that item is already selected.

In LispWorks 5.0 and previous versions, the selection callback is not called in the reselection case.

12.4.3 tree-view-update-an-item

capi:tree-view-update-an-item is deprecated.

Please use **capi:tree-view-update-item** instead, which behaves in just the same way.

12.5 New graphics ports features

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

12.5.1 Finding all available font names

The new function **gp:list-all-font-names** returns a list of partially-specified font description objects for each available font.

12.5.2 Transparency on X11/Motif

Some image formats with transparency are now supported on X11.

When using **gp:draw-image** on X11, graphical images loaded from files (for example some GIFs and some TIFFs) are now drawn using the transparency mask from the image.

In LispWorks 5.0 and previous versions, this only worked on Cocoa and Microsoft Windows.

12.6 More new features

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

12.6.1 LispWorks as a dynamic library

You can now save or deliver 32-bit LispWorks as a dynamic library (**.so** or **.dylib** files) on Intel Macintosh, Linux and FreeBSD, and you can save or deliver 64-bit LispWorks as a dynamic library on Intel Macintosh and Linux.

This functionality was already available (creating **.dll** files) in LispWorks 5.0 for Windows and earlier versions.

For details of creating a LispWorks dynamic library see the **:dll-exports** and **:dll-added-files** arguments to **save-image** and **deliver**. For a description of the behavior and the API provided see the section "LispWorks as a dynamic library" in the *LispWorks User Guide*.

12.6.2 Relocation improvements

The interface allowing most LispWorks implementations to be relocated with a different base address has been extended to include LispWorks dynamic libraries.

For relocation of an executable, the argument **--relocate-image** is no longer required to be the first command line argument.

It is now also possible to specify a different reserved range when relocating the image on most platforms.

See "Startup relocation" in the *LispWorks User Guide* for details.

12.6.3 Larger heaps in LispWorks (32-bit) for Windows

When growing large (more than the initially reserved 0.5GB) LispWorks 5.1 (32-bit) for Windows "skips" over address space occupied by DLLs which it cannot use. This allows it to grow above the libraries and thus achieve a larger maximum heap. The size depends on the mapping of the libraries.

LispWorks 5.0 (32-bit) for Windows and previous versions requires a contiguous heap and thus its heap is limited in practise to less than 1GB.

For more information about the memory layout of LispWorks see "Startup relocation" in the *LispWorks User Guide*.

12.6.4 Stream locking

LispWorks now has "cooperative" stream locking. "Cooperative" here means that the lock is not explicitly claimed by the stream I/O functions, so the locking takes effect only when your code uses the locking interface explicitly ("cooperates").

Stream locking makes it possible to write atomically with respect to code that runs in another thread, providing that the code running in the other thread also uses the locks. The same functionality can be achieved by using a **`mp:lock`** around I/O calls, but stream locking saves you from keeping a **`mp:lock`** object for each stream.

Stream locking is an experimental feature and is not documented in the manuals supplied. For more information please contact Lisp Support.

12.6.5 Telling LispWorks about your own defining forms

You can now extend the Dspec system to know about new dspec classes and their defining forms.This allows commands such as **Find Source** and **Undefine** to operate correctly on your definitions.

See "Dspec classes" in the *LispWorks User Guide*.

12.6.6 Windows registry API

LispWorks for Windows now supports functions for accessing, creating and deleting registry entries. For an introduction see the Operating Environment chapter in the *LispWorks User Guide*.

Compatibility Note: Mostly we have simply documented functionality that exists (undocumented) in LispWorks 5.0. However two functions have changed, as follows:

In **win32:create-registry-key** the *access* parameter (was *security*) now supports integer values. *key* parameter has been renamed *handle* to avoid ambiguity with *subkey* which is a string.

win32:close-registry-key now has an *errorp* argument and *expected-type* **:string** handler for **:environment-string** to return the raw string.

12.6.7 Optimize your code with compiler explanations

The **:explain** declaration controls messages printed by the compiler while it is processing forms, helping you to optimize your code.

Various keywords allows you to see information about compiler transformations depending on type information, allocation of floats and bignums, floating point variables, function calls, argument types and so on. See the section "Optimizing your code" in the *LispWorks User Guide*.

12.6.8 Extended profiling API

A new profiler mode allows programmatic control of profiling, including control over which processes are profiled. See the section "Running the profiler" in the *LispWorks User Guide*.

Extensions to **hcl:set-up-profiler** add more control over profiling and the output.

There are also new APIs to print profile call trees, see **hcl:profiler-tree-from-function** and **hcl:profiler-tree-to-function**.

12.6.9 External format converters

Utilities for converting between a Lisp string and an encoded byte vector are improved and documented. See **ef:encode-lisp-string** and **ef:decode-external-string**.

12.6.10 Chinese encoding supported

Chinese Simplified (GBK) encoding is now supported. Use *external-format* **:gbk** when calling **cl:open** or using strings with the Foreign Language Interface.

12.6.11 OpenSSL interface extended

See **comm:set-verification-mode**, **comm:get-verification-mode**, **comm:openssl-version** in the *LispWorks Reference Manual* and various direct FLI definitions for OpenSSL listed in the *LispWorks User Guide*.

12.6.12 SO_KEEPALIVE and TCP_NODELAY

The socket stream interface now supports setting of SO_KEEPALIVE and TCP_NODELAY on the socket. See **comm:open-tcp-stream** and **comm:start-up-server**.

12.6.13 Memory management helper after loading

hcl:finish-heavy-allocation is a new function which may be useful after your application creates many long-lived objects, for example after loading large amounts of code or data.

hcl:with-heavy-allocation is now implemented in 64-bit LispWorks.

12.6.14 64-bit memory management API extended

system:set-memory-check and **system:set-memory-exhausted-callback** are new in 64-bit LispWorks.

12.6.15 Accessing symbol value cells across processes

A new function **mp:symeval-in-process** gets (or with **setf**, sets) the value of symbol which is dynamically bound in a given process.

12.6.16 Inspector command to display the rest of the current object.

The new inspector command **:dr** displays the rest of the current object. It is useful when you already used **:dm** to display a limited number of slots and then decide to display all the remaining slots of the object. See the Debugger chapter in the *LispWorks User Guide*.

12.7 IDE changes

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

12.7.1 Compilation output highlights warnings and errors

Warnings and errors signalled during compilation are now highlighted in the Output tab of the Editor and System Browser tools. Pressing the **Return** key allows you to view these conditions in a Compilation Conditions Browser tool.

You can use the editor command **Edit Recognized Source** (key **Ctrl-x ,** in Emacs emulation) or the context menu to jump to the source when the cursor is inside the highlighted region.

12.7.2 Compilation Conditions Browser improved

The Compilation Conditions Browser tool is simplified and easier to use.

The Conditions tree shows the conditions signalled in each file along with any compiler explanations obtained via the **:explain** declaration. Also, the tool is easy to reach from the Editor or System Browser tool's Output tab as described in "Compilation output highlights warnings and errors" on page 115.

12.7.3 Dragging files to the Editor

On Cocoa, you can now drag a file from another application (such as the Finder) and drop it into the LispWorks Editor tool.

On Microsoft Windows, you can drag a file from another application (such as the Windows Explorer) and drop it into the LispWorks Editor tool. To enable this, you need to switch on the option **Tools > Global Preferences.... > Respond to drag and drop**.

12.7.4 Control of prompting on exit

By default, the LispWorks IDE prompts for confirmation on exiting. You can control this behavior so that it never prompts, or prompts only when there are unsaved editor buffers, by setting the **Confirm Before Exiting** option as described in the *Common LispWorks User Guide*.

The option is effective when you quit LispWorks by any UI gesture, including the Emacs key **Ctrl+X Ctrl+C**, the Windows key **Alt+F4**, the Macintosh key **Command+Q**, or menu commands like **LispWorks > Quit**, **File > Exit**, **Works > Exit > LispWorks.**

12.7.5 Controlling the floating toolbar on Cocoa

On Cocoa, by default a floating window offers one-click access to the various IDE tools.

If you do not want to see this, you can now switch it off by deselecting the option **LispWorks > Preferences... > Show the tools on a floating toolbar**.

12.7.6 Hiding toolbars

You can now choose whether IDE tools display a toolbar containing buttons for commonly-used operations.

If you do not want to see the toolbar for a particular tool, switch it off by deselecting the option **Tools > Preferences... > General > Show Toolbar**.

12.7.7 Recent Files menu

The **File** menu now has a **Recent Files** submenu which lists the last 10 files visited via the **File > Open...** and **File > Save As...** commands.

12.7.8 Listener File menu commands

The Listener tool now offers **File > Save As...** menu command, which prompts for a path and saves the Listener buffer's contents.

Also, **File > Open...** now remembers the last path after program restart.

12.7.9 Accelerator keys for tools

Commonly-used tools in the IDE now have accelerators, which are available in some editor emulations. For details see "Displaying tools using the keyboard" in the *Common LispWorks User Guide*.

12.7.10 In-place completion in text input panes

Text input panes such as the **Class:** pane of the Class Browser now support in-place completion allowing faster selection from a list of possible inputs. This behaves as in the editor, described under "In-place completion" on page 120.

12.7.11 Controlling completion behavior

In-place completion is enabled by default in the IDE.

If you prefer a modal dialog style of completion, deselect the **Use in-place completion** option. You can find this option on Microsoft Windows and Motif by **Tools > Global Preferences...** and on Cocoa by **LispWorks > Preferences...**.

12.7.12 Improved filtering of lists

The Filter area of tools which display lists has been improved.

The items in the list are now filtered as soon as you type in the filter box and there is no need to make a separate confirmation gesture. Therefore the confirm button (with a green tick image) has been removed.

The filter modes button now indicates which of the three independent filter modes **Regexp Search**, **Exclude Matches** and **Case Sensitive** are in effect, and new keyboard gestures allow switching these modes. The keys are **Ctrl+Shift+R**, **Ctrl+Shift+E** and **Ctrl+Shift+C**.

Similar filter functionality is also available in completion lists.

12.7.13 Debugger display of frames and arguments

The Debugger tool can now display the frames and the arguments of the current frame in two list panels, rather than the default view, which is a tree. Raise the Debugger's **Preferences...** dialog to change this option.

12.7.14 Pasting Clipboard objects

Edit > Paste in an editor-pane-based tool now pastes the printed representation of a Clipboard object.

12.7.15 Defining Stepper breakpoints

The **Conditional** and **Printing** breakpoint forms are now read in the package where the stepped function was defined. This package is displayed in the **Conditional** and **Printing** tabs of the **Edit breakpoints** dialog.

12.7.16 Colors in the Symbol Browser

On Microsoft Windows the Symbol Browser tool now uses color to indicate the types of definitions on the symbols it finds. See the *Common LispWorks User Guide* for details.

12.7.17 Hiding accessors in the Class Browser

A new option **Include Accessors** in the **Functions** view of the Class Browser allows you to control whether it includes or excludes accessor methods in the list of functions displayed.

12.7.18 System Browser graph menus

The System Browser tool's graph context menu is now dependent on the selection. If a system is selected, system operations such as **Concatenate** and **Search Files** are shown. If a file is selected, file operations such as **Browse Parent System** and the **Recent Files** submenu are shown, and the system operations are on the **Systems** submenu.

This context menu no longer offers an **Edit** submenu because it was not useful and inconsistent with most other tools.

12.7.19 Preferences keyboard shortcut

On Cocoa the standard shortcut **Command+,** for the application **Preferences...** dialog is now supported.

12.7.20 Native editor emulation improvements

12.7.20.1 Cursor width

The Editor tool's cursor is now just one pixel wide in Microsoft Windows and Macintosh editor emulations.

12.7.20.2 Highlighting of Lisp forms

Highlighting of Lisp forms has been improved to take account of cursor behavior on scroll in native emulations. See "Matching parentheses are both highlighted" on page 122.

12.7.20.3 KDE/Gnome keys

Native emulation on Motif now gives KDE/Gnome editor keys including for example:

```
Control+W      Delete Window

Control+Q      Save all Files and Exit
```

12.7.20.4 Enter key evaluates on Macintosh

This is a new key defined in Macintosh editor emulation:

```
Kp-Enter        Evaluate Defun
```

12.8 Editor changes

See the *LispWorks Editor User Guide* for details of these changes.

12.8.1 Locking changed

The editor now uses fine-grained locking rather than process-level locking using **mp:without-interrupts** and **mp:without-preemption**. If you implement editor extensions, you should now use the macros **editor:with-buffer-locked** and **editor:with-point-locked** to get locking.

12.8.2 Undo changed

The implementation of undo functionality has changed significantly. If you implement editor extensions based on the supplied source code, you should review these by comparing with the editor source code supplied with LispWorks 5.1. In particular, use **editor:collect-undo** rather than **recording-for-undo** and ***dont-undo***.

12.8.3 In-place completion

Editor commands which help you to complete what you are typing, such as **Complete Symbol** (key **Alt+Ctrl+I** in Emacs emulation) and **Complete Input** (key **Tab** in the echo area) now display a window listing possible completions which does not grab the input focus.

While this in-place completion window is displayed, most keyboard input is still processed by the editor as normal, so you can simply continue typing to reduce the number of possible completions displayed.

A few keyboard gestures are handled by the completion window to allow you to select an item from the list of possibilities. These are **Up**, **Down**, **PageUp**, **Page-Down** and **Return**, while **Escape** cancels the in-place window.

Occasionally you may want to filter the completions list. **Ctrl+Return** adds/removes this filter. **Ctrl+Shift+Return** redirects input to the filter. **Ctrl+Shift+R**, **Ctrl+Shift+E** and **Ctrl+Shift+C** change the filter mode.

In-place completion allows faster completion, when you get used to it! If you prefer modal completion dialogs instead (as in LispWorks 5.0 and earlier versions) change the **Use in-place completion** setting as described in "Controlling completion behavior" on page 117.

12.8.4 New completion command

The new editor command **Abbreviated Complete Symbol** is bound to **Meta+Shift+I** in Emacs emulation. Enter for example **w-o-f** and complete to **with-open-file**.

Abbreviated Complete Symbol uses the in-place completion UI.

12.8.5 New location commands

The new editor commands **Go Back** and **Select Go Back** allow you to return to previously recorded locations in editor buffers. They are bound to **Ctrl+X C** and **Ctrl+X M** in Emacs emulation.

Locations are recorded automatically by the editor for most commands that take you to a different buffer or where you might lose your place within the current buffer. They are designed to be a more comprehensive form of the mark ring, but without the interaction with the selected region.

12.8.6 Feedback on key bindings for commands

The editor now tells you about command key bindings that you do not use. When you invoke an editor command via **Extended Command** and key bindings exist for that command, a message is displayed in the echo area listing the available keys.

You use **Extended Command** when you do **Meta+X** *command name* in Emacs emulation.

12.8.7 Editor defaults to Code Page encoding

On Windows files are now written using the Code Page encoding when when an encoding is not specified (for instance by setting the Editor File Encodings Output preference to DEFAULT). Files are read using the Code Page when an encoding is not specified. In LispWorks 5.0 and earlier, Latin-1 was used but this does not handle non base characters such as the Trademark symbol.

12.8.8 Encoding for writing files

The editor variable **`Output-Format-Default`** now has initial value **`nil`**. In LispWorks 5.0 and earlier versions it was a Latin-1 encoding, which does not support certain characters obtained from other applications. Now the encoding to use when first saving a file is chosen essentially as if by calling **`open`**, although on Microsoft Windows code pages similar to Latin-1 are never mapped to Latin-1.

The main effect of this change is that, with the default configuration of **`open`** on Windows, buffers containing non-Latin-1 characters that are in the system code page can be saved to file without the need to specify an external format.

Set the value of **`Output-Format-Default`** if you want a default encoding for writing files in the editor.

12.8.9 Encoding for reading files

Now the encoding used when reading a file (for example by **File > Open...** or **`Ctrl+X Ctrl+F`**) is chosen essentially as if by calling **`open`**.

The main effect of this change is that, with the default configuration of **`open`** on Windows, files which do not somehow specify an encoding will be read in the system code page encoding.

Set the value of **`Input-Format-Default`** if you want a default encoding for reading files in the editor.

12.8.10 Matching parentheses are both highlighted

In Lisp mode, both parentheses delimiting a form are highlighted when the cursor is at one of them. The new behavior is particularly useful in Microsoft

Windows emulation and Macintosh editor emulation, because the cursor can be outside the visible region. In LispWorks 5.0 and previous versions only the parenthesis opposite the cursor position is highlighted.

The highlight uses a light green background by default. Its appearance is controlled by the Editor Style preference named "Show Point".

12.8.11 Highlighting of the selected region is removed after scrolling

In Emacs emulation the selection now becomes unhighlighted when the cursor moves due to scrolling (that is, the cursor would have moved out of the visible region and thus gets moved automatically). In LispWorks 5.0 the altered highlight is confusing to some users, and is rarely useful.

12.8.12 Editor filling and autofilling of comments in Lisp mode

The **Fill Paragraph** command in Lisp Mode (**Alt+Q**) now correctly detects comments (lines starting with more or more semicolon) and fills them using the same number of semicolons as the fill prefix.

Filling of comments triggered in Auto Fill Mode now uses the spaces surrounding the comment characters for each subsequent filled line as well.

12.8.13 Syntax of editor variable names

Editor variable names are now symbols rather than strings. For example, where you previously used:

```
(editor:variable-value "Add Newline At Eof On Writing File")
```

you should now use:

```
(editor:variable-value 'editor:add-newline-at-eof-on-writing-
file)
```

12.9 Foreign Language interface changes

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

12.9.1 Accessing 64-bit integer types

In 64-bit LispWorks **fli:foreign-typed-aref** now supports **(signed-byte 64)** and **(unsigned-byte 64)**.

12.9.2 Slots in nested structures

You can now access slots inside nested foreign structures, by passing a list *slot-name* argument to functions like **fli:foreign-slot-value**.

12.9.3 More options for dynamic foreign objects

fli:with-dynamic-foreign-objects now allows you to allocate arrays of objects, and also allows more complex initialization of objects. See the description of its *bindings* argument.

12.9.4 Dereferencing null pointer is an error

In LispWorks 5.0 and previous versions, calling **fli:dereference** with a null pointer would return **nil**. In LispWorks 5.1 it now signals an error. You can use **fli:null-pointer-p** to detect null pointers.

This change does not affect **(setf fli:dereference)** which has always signaled an error with a null pointer.

12.9.5 Defining opaque pointers

The new macro **fli:define-opaque-pointer** defines a foreign pointer type and structure type where there is no structure description, as in

```
typedef struct structure-type *pointer-type;
```

It is useful for automatic type checking of foreign pointers returned and passed to another foreign function.

12.9.6 Byte packing

Byte packing for C structures from Lisp is now documented. See **fli:define-c-struct**.

12.9.7 Protected dynamic foreign execution

The new macro **fli:with-dynamic-foreign-objects-with-cleanups** combines the effects of **fli:with-dynamic-foreign-objects** and **unwind-protect**.

12.10 COM/Automation changes

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

12.10.1 Calling property getters

com:call-dispatch-method and **com:invoke-dispatch-method** can now be used with property getters. For details see the "Calling Automation methods" section of the manual.

12.10.2 Implementing COM interfaces not in the type library

The **:extra-interfaces** class option of **com:define-automation-component** allows you to specify COM interfaces that the object will implement in addition to the interfaces implied by the **:coclass** option. This allows the object to implement other interfaces not mentioned in the type library.

12.10.3 New function com-object-from-pointer

com:com-object-from-pointer returns the COM object that implements a particular COM interface pointer.

12.11 Common SQL changes

12.11.1 Setting Oracle connection parameters

There are improved default prefetch values for Oracle database connections, and control over the amount to prefetch. See "Setting connection parameters" in the *LispWorks User Guide*.

12.11.2 Calling SQL infix operators

Calling infix operators via Common SQL is now documented. See the pseudo operators **sql-operator** and **sql-boolean-operator** under **sql:sql-operation** in the *LispWorks Reference Manual*.

12.11.3 Better handling of LIKE

Unnecessary checks in the processing of the SQL operator **LIKE** via Common SQL have been removed. It now accepts arbitrary functions.

12.12 Application delivery changes

See the *LispWorks Delivery User Guide*.

12.12.1 Dynamic libraries

You can now deliver a dynamic library (**.so** or **.dylib**) as described in "LispWorks as a dynamic library" on page 111.

12.12.2 Default location of target application bundle

When using the **configuration/save-macos-application.lisp** example code to deliver a Cocoa application (or simply to save a new image), the new application bundle is now written into the user's home directory.

In LispWorks 5.0 the default location was alongside the LispWorks application bundle, a location which may not be writable.

To write your application bundle in another location, use **executable-application-bundle-directory** as illustrated in **configuration/save-macos-application.lisp**.

12.12.3 Vista icons

The **deliver** keyword **:icon-file** now supports 256x256 icons on Windows Vista. PNG compressed icons are supported.

12.12.4 Save modified buffers action removed

The action item "Save modified buffers" no longer exists. Therefore you should no longer call

```
(lw:undefine-action "Confirm when quitting image" "Save modified
buffers")
```

12.12.5 Command line for delivery

The use of **-init** command line argument to run LispWorks with a **deliver** script is deprecated. Use **-build** instead, or use the Application Builder tool in the LispWorks IDE.

12.12.6 Quitting LispWorks DLLs

You can now make a LispWorks DLL quit gracefully so that its main application can unload it.

See the section "Delivering a DLL" in the *LispWorks Delivery User Guide*.

12.12.7 Runtime generation for universities and colleges

LispWorks Academic Site Edition is no longer built as a separate product. Academic site license customers will be upgraded to LispWorks 5.1 licenses with all the functionality of LispWorks 5.0 Academic Site Edition plus application delivery (that is, generation of runtimes).

12.13 CLOS/MOP changes

12.13.1 Forward referenced class and type predicates

Type predicates **typep** and **subtypep** now work when passed names of forward referenced classes. In LispWorks 5.0 and previous versions, this would signal an error. Thanks to Pascal Costanza for pointing out that ANSI Common Lisp 4.3.7 and AMOP both state that forward reference classes have proper names.

12.13.2 slot-definition changes for AMOP compatibility

clos:slot-definition now records a supplied *allocation* initarg. **clos:slot-definition-allocation** is now a normal accessor function.

clos:slot-definition now contains the *documention* slot. This was defined only in its subclass **clos:standard-slot-definition** in LispWorks 5.0 and previous versions.

12.14 CLIM changes

12.14.1 accepting-values bug fix

A bug causing an error when ther user clicks a button created by **accept-values-command-button** inside **accepting-values** is fixed.

12.15 Other changes

12.15.1 Changes in *features*

cl:*features* contains **:lispworks5.1** and **:lispworks5**. It no longer contains **:lispworks5.0**.

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 Reference Manual*.

12.15.2 New fasl filename extension

The default fasl filename extension in 64-bit LispWorks for Macintosh is **64nfasl** on the PowerPC architecture and **64xfasl** on Intel.

Fasl file types for the various LispWorks platforms are documented under **compile-file** in the *LispWorks Reference Manual*.

12.15.3 Loading old data files

Binary files created with **hcl:dump-forms-to-file** or **hcl:with-output-to-fasl-file** in LispWorks 5.0, LispWorks 4.4 or LispWorks 4.3 can be loaded into LispWorks 5.1 using **sys:load-data-file**.

Note: because the default fasl extensions have changed on some platforms since LispWorks 4.4 you may need to do something like this to load old binaries:

```
(let ((sys:*binary-file-type* "fsl"))
  (sys:load-data-file "C:/lww44.fsl"))
```

12.15.4 Loading logical pathnames

When **load** is given a logical pathname, it now uses the translated pathname when determining whether the file is source code or a fasl. That is, if

```
(translate-logical-pathname "HOST:FOO;bar.BIN")
=>
#P"W:\\Development\\foo\\bar.nfasl"
```

then **load** will treat **"HOST:FOO;bar.BIN"** as a fasl on platforms where **nfasl** is fasl extension.

In LispWorks 4.4 and previous versions, **load** can incorrectly load it as a text file because **.BIN** was not recognised as a binary file type.

12.15.5 Changes to cl:directory

cl:directory now works correctly when the directory component of its *pathspec* argument contains **:wild-inferiors** followed by further directories. Such a *pathspec* can be constructed by parsing namestrings like **"**/foo/*.lisp"**.

Also, **cl:directory** can now match the **file-namestring** of its directory argument as a 'flat' string, rather than a name and type. To get this behavior supply a true value for the new keyword argument **:flat-file-namestring**.

12.15.6 get-folder-path and get-user-profile-directory changed

sys:get-folder-path and **sys:get-user-profile-directory** now return a directory pathname object that represents the directory. In LispWorks 5.0 and previous versions they both return a string. This change makes it easier to construct a pathname for a file in the directory, for example:

```
(make-pathname :name "foo"
               :type "lisp"
               :defaults (sys:get-folder-path :my-documents))
```

12.15.7 Reader handles Unicode BOM

A Unicode (UCS-2 encoded) file beginning with the Byte Order Mark (BOM, Lisp character **#\U+FEFF**, also known as Zero-Width-No-Break-Space) can now be read by the Lisp reader. The standard lisp readtable now treats this character as whitespace.

This character is written at the start of a UCS-2 encoded file by some programs including Microsoft Notepad and the LispWorks editor. This change allows **read** to work on such files without specially ignoring the BOM.

12.15.8 Array constant values

array-total-size-limit is now correctly set to be the smallest limit for any element type as per the ANSI Common Lisp standard. Its value is 2^26 in 32-bit LispWorks, and 2^29 -1 in 64-bit LispWorks.

array-dimension-limit is also set to be the smallest limit for any element type. Its value is almost 2^26 in 32-bit LispWorks, and 2^29 - 1 in 64-bit LispWorks.

12.15.9 Accuracy and correctness of floating-point format directives

There are various fixes to the floating-point **format** directives **~F**, **~E**, **~G** and **~$**.

~E and **~G** now print the number as accurately as **~F**.

Exponential notation prints the correct number of digits before the decimal point for all exact powers of 10 and when rounding from 9 to 10 causes the number of digits to change.

~E prints the exponent in decimal regardless of ***print-base***.

~F, **~E**, **~G** and **~S** print negative zero with a leading minus sign.

12.15.10 Parsing floats

The function **parse-float** is now documented. It parses a floating-point number from a string and returns a float object. See the *LispWorks Reference Manual* for details.

12.15.11 define-modify-macro more strict

The macro **define-modify-macro** now signals an error if the *function* argument is not a symbol, as specified by the ANSI Common Lisp standard.

12.15.12 Setting file times

The function **system:set-file-dates** which sets the modification and access times of a file is now documented in the *LispWorks Reference Manual*.

12.15.13 Implementing cl:file-position for user-defined stream classes

The generic function **stream:stream-file-position** is now documented in the *LispWorks Reference Manual*.

12.15.14 File encoding guesser improved

sys:guess-external-format (called by **open** for character file streams) no longer overrides any specified external format name that it is given (via the **:external-format** argument to **open**).

Also **sys:merge-ef-specs** has been changed to merge the parameters only in cases where it can guarantee correctness.

See the *LispWorks Reference Manual* for details of both these functions.

12.15.15 Command line for save-image scripts

The use of **-init** command line argument to run LispWorks with a **save-image** script is deprecated. Use **-build** instead, or use the Application Builder tool in the LispWorks IDE.

12.15.16 setup-for-alien-threads removed

The function **system:setup-for-alien-threads** no longer exists. LispWorks for Windows now automatically performs the operations that are needed to cope with foreign callbacks from unknown threads.

12.15.17 add-special-free-action accepts function designators

The *function* argument to **hcl:add-special-free-action** can be any function designator, not only a symbol.

12.15.18 Data Execution Protection and LispWorks DLLs

LispWorks for Windows DLLs now run when the Data Execution Protection (DEP) is set at the highest level. The DEP feature was introduced in Windows XP sp2/Windows Server 2003 sp1.

DLLs built with LispWorks 5.0 and previous versions fail to run at this setting, requiring the application to be added to the list of allowed applications.

Note: LispWorks for Windows executables run regardless of the DEP setting.

12.15.19 Multiprocessing API changes

The **:lock-name** argument to **mp:make-mailbox** is deprecated and should no longer be used. **mp:make-lock** has been revised while **mp:lock-lock** and **mp:lock-count** are no longer supported.

The functions **mp:claim-lock** and **mp:release-lock** are no longer available. Use **mp:process-lock** and **mp:process-unlock** instead.

The function **mp:process-read-event** is deprecated. Please use the new documented function **mp:process-wait-for-event** instead.

The function **mp:schedule-named-event** is deprecated and no longer exported. Please use **mp:make-timer** and **mp:schedule-timer-relative** instead.

The functions **mp:process-stop** and **mp:process-unstop** have been revised; **mp:process-stopped-p** and **mp:process-continue** are new.

Use of **(setf mp:process-arrest-reasons)** is deprecated and its behavior differs from LispWorks 5.0 and previous versions. Please use **mp:process-stop** instead.

mp:get-process and **mp:get-current-process** are new.

12.15.20 IDE tool icons removed

The tool-specific icons for Listener, Editor and Inspector windows in LispWorks for Windows have been removed.

12.15.21 LessTif no longer supported

LispWorks for Linux no longer supports LessTif. Use OpenMotif as described in the Installation Guide.

12.15.22 DDE error class

win32::dde-error now inherits from **cl:error**.

In LispWorks 5.0 and previous versions, it inherited from **cl:condition**.

12.15.23 debugger :trap in 64-bit LispWorks

The debugger **:trap** command now works on 64-bit platforms. It was previously only implemented for 32-bit LispWorks.

12.15.24 Objective-C protocols

It is not possible to define new protocols entirely in Lisp on Mac OS X 10.5, but **objc:define-objc-protocol** can be used to declare existing protocols.

12.16 Documentation changes

The MP package is now better documented in the *LispWorks Reference Manual*.

There are no printed manuals available for LispWorks 5.1. Printable versions of all the manuals are available at **www.lispworks.com/documentation/**.

12.17 Binary Incompatibilty

If you have binaries (fasl files) which were compiled using LispWorks 5.0 or previous versions, please note that these are not compatible with this release. Please recompile all your code with LispWorks 5.1

12.18 Known Problems

12.18.1 Runtime library requirement on Windows

LispWorks for Windows requires the Microsoft Visual Studio runtime library **msvcr80.dll**. The LispWorks installer installs this DLL if it is not present.

Applications that you build with LispWorks for Windows also require this DLL, so you must ensure it is available on target machines. It is part of Windows Vista, but for earlier Windows operating systems you should use the Microsoft redistributable mentioned below.

At the time of writing, the redistributable **vcredist_x86.exe** for use with for LispWorks (32-bit) applications is freely available at

http://www.microsoft.com/downloads/details.aspx?familyid=32BC1BEE-A3F9-4C13-9C99-220B62A191EE&displaylang=en

The redistributable **vcredist_x64.exe** for use with LispWorks (64-bit) applications is freely available at

http://www.microsoft.com/downloads/details.aspx?FamilyID=90548130-4468-4bbc-9673-d6acabd5d13b&DisplayLang=en

Run the redistributable from your application's installer, or tell your users to run it directly themselves before running your application.

Note that Windows Installer 3.0 or later is also needed on the target machine for Windows 2000 or XP.

12.18.2 Problems with LispWorks for Macintosh

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

Functions run by **mp:process-run-function** have their standard streams connected to ***terminal-io*** (which is not normally visible). Possibly when the IDE is running, output should be connected to the Background Output buffer.

Reading from ***terminal-io***, closing Terminal.app and then reading again gets end of file.

12.18.3 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 doesn't work for errors in Cocoa Event Loop or Editor Command Loop threads. However, there is a **Get Backtrace** button so you can obtain a backtrace and also a **Debug Snapshot** button which aborts from the error but displays a debugger with a copy (snapshot) of the stack where the error occurred.

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

Setting ***enter-debugger-directly*** to **t** can allow the undebuggable processes to enter the debugger, resulting in the UI freezing.

Inspecting a long list (for example, 1000 items) via the Listener's **Inspect Star** editor command prompts you about truncation in a random window. If you cancel, the inspect is still displayed.

The editor's Help about help (**Control+H Control+H**) dialog is messy because it assumed that a fixed width font is being used.

It is impossible to interrupt loops in the Cocoa Event Loop process.

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.

12.18.4 Problems with CAPI and Graphics Ports on Cocoa

Some graphics state parameters are ignored, in particular operation, stipple, pattern, fill-style and mask (other than a rectangle).

LispWorks ignores the System Preferences setting for the smallest font size to smooth.

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.

Programmatic scrolling of **capi:list-panel** etc is not implemented.

Caret movement and selection setting in **capi:text-input-pane** is implemented, but note that it works only for the focussed pane.

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

There is no meta key in the input-model of **capi:output-pane**. Note that, in the editor when using Emacs emulation, the **Escape** key can be used as a prefix.

There has been no testing with 256 color displays.

There is no visual feedback for dead-key processing, for example **Option+n** is the tidle dead-key.

The graph pane's plan mode rectangle doesn't redraw when moved or resized.

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

All menu items are disabled when a dialog is on the screen. However, the **Command+X**, **Command+C** and **Command+V** shortcuts work within text panes

There is no way to make the close icon on a window show the "modified" state (**NSWindow:setDocumentEdited**).

capi:editor-pane will only work with fonts whose widths are (almost) integral for example Monaco 10, 15, 20 pt etc but not Monaco 12 pt. The nearest good size is used instead.

The default menu bar is visible when the current window has no menu bar.

capi:tree-view is slow for a large number (thousands) of items.

The editor displays decomposed characters as separate glyphs.

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

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

12.19 Recyclable packaging

If you have received LispWorks 5.1 on a CD-ROM, please note that the packaging is 100% paper-recyclable and biodegradable. Please recycle the LispWorks CD-ROM case when you are done with it.

Index

A

accepting-values macro 128
accessors
  slot-definition-allocation 128
ActiveX 106
add-special-free-action function 132
:after-input-callback initarg 108
array-dimension-limit constant 130
array-total-size-limit constant 130

B

:before-input-callback initarg 108

C

call-dispatch-method macro 125
call-editor generic function 108
CD-ROM case
  biodegradable 137
  recyclable 137
CD-ROM digipack
  biodegradable 137
  recyclable 137
:change-callback initarg 108
classes
  cocoa-default-application-interface 107
  docking-layout 136
  editor-pane 108, 137
  filtering-layout 107
  interface 110
  list-panel 107, 110, 136
  multi-column-list-panel 137
  ole-control-component 106
  opengl-pane 110
  option-pane 110
  output-pane 136
  simple-pane 106
  slot-definition 128
  standard-slot-definition 128
  switchable-layout 108
  tab-layout 108
  text-input-pane 109, 136
  tree-view 107, 136, 137
close-registry-key function 113
cocoa-default-application-interface class 107
collect-undo macro 120
color
  in list items 107
  in the Symbol Browser 118
:color-function initarg 107
:combine-child-constraints initarg 108
command line arguments
  -build 127
  -init 127
  --relocate-image 111
com-object-from-pointer function 125
compile-file function 106
Confirm Before Exiting 116
constants
  array-dimension-limit 130
  array-total-size-limit 130
create-registry-key function 113

D

Data Execution Protection 132
decode-external-string function 114
define-automation-component macro 125
define-c-struct macro 124
define-interface macro 110
define-modify-macro macro 131
define-objc-protocol macro 133
define-ole-control-component macro 106
define-opaque-pointer macro 124
deliver function 106, 111, 126, 127
DEP 132
dereference function 124
directory function 129
display-dialog function 137
docking-layout class 136
:dock-menu initarg 107
drag and drop 106, 116
drag-pane-object function 106
draw-image function 111
draw-metafile function 106
:drop-callback initarg 106
dump-forms-to-file function 129

E

editor-pane class 108, 137
encode-lisp-string function 114
enter-debugger-directly variable 135
execute-with-interface-if-alive function 109
exit
  IDE behavior 116
:explain declaration 113, 115
extended-time macro 99

F

Failed to enlarge memory 88
dont-undo variable 120
features variable 128
filtering lists
  in CAPI windows 107
  in completion windows 121
  in tools 117
filtering-layout class 107
finish-heavy-allocation function 114
foreign callbacks 132
foreign-slot-value function 124
foreign-typed-aref function 124
format directives 130
format function 130
functions
  add-special-free-action 132
  close-registry-key 113
  com-object-from-pointer 125
  compile-file 106
  create-registry-key 113
  decode-external-string 114
  deliver 106, 111, 126, 127
  dereference 124
  directory 129
  display-dialog 137
  drag-pane-object 106
  draw-image 111
  draw-metafile 106
  dump-forms-to-file 129
  encode-lisp-string 114
  execute-with-interface-if-alive 109
  finish-heavy-allocation 114
  foreign-slot-value 124
  foreign-typed-aref 124
  format 130
  get-current-process 133
  get-folder-path 130
  get-process 133
  get-user-profile-directory 130
  get-verification-mode 114
  guess-external-format 131
  invoke-dispatch-method 125
  list-all-font-names 110
  load 106, 129
  load-data-file 129
  load-icon-image 107
  make-lock 132
  make-mailbox 132
  make-timer 133
  merge-ef-specs 131
  null-pointer-p 124
  open 114, 122
  openssl-version 114
  open-tcp-stream 114
  parse-float 131
  process-arrest-reasons 133
  process-continue 133
  process-lock 132
  process-read-event 132
  process-run-function 135
  process-stop 133
  process-stopped-p 133
  process-unlock 132
  process-unstop 133

process-wait-for-event 132
profiler-tree-from-function 113
profiler-tree-to-function 113
prompt-for-file 108
read 130
room 99
save-image 12, 106, 111, 132
schedule-named-event 133
schedule-timer-relative 133
screen-active-interface 109
set-file-dates 131
set-memory-check 114
set-memory-exhausted-callback 114
set-object-automatic-resize 108
setup-for-alien-threads 132
set-up-profiler 113
set-verification-mode 114
set-window-split-on-space 109
sql-operation 126
start-environment 11, 84
start-up-server 114
subtypep 127
symeval-in-process 115
translate-logical-pathname 129
tree-view-update-an-item 110
tree-view-update-item 110
typep 127

G

Garbage Collector message 88
Garbage Collector output 88
GC message 88
GC output 88
generic functions
call-editor 108
stream-file-position 131
get-current-process function 133
get-folder-path function 130
get-process function 133
get-user-profile-directory function 130
get-verification-mode function 114
guess-external-format function 131

I

IDE tools
accelerator keys 117
Class Browser 118
Compilation Conditions Browser 115
Debugger 117, 118
Editor 116
filtering lists in 117
floating toolbar 116
hiding the toolbar 116
Listener 117
Preferences dialog 119
Stepper 118
Symbol Browser 118
System Browser 119
in-place completion 117
interface class 110
invoke-dispatch-method function 125

L

LIKE SQL operator 126
:line-wrap-face initarg 109
:line-wrap-marker initarg 109
LispWorks as ActiveX control 106
list-all-font-names function 110
list-panel class 107, 110, 136
load function 106, 129
load-data-file function 129
load-icon-image function 107

M

macros
accepting-values 128
call-dispatch-method 125
collect-undo 120
define-automation-component 125
define-c-struct 124
define-interface 110
define-modify-macro 131
define-objc-protocol 133
define-ole-control-component 106
define-opaque-pointer 124
extended-time 99
profile 99
recording-for-undo 120
with-buffer-locked 120
with-dynamic-foreign-objects 124
with-dynamic-foreign-objects-with-cleanups 125
with-external-metafile 106
with-heavy-allocation 114
without-interrupts 120
without-preemption 120
with-output-to-fasl-file 129
with-page 136
with-point-locked 120
make-lock function 132
make-mailbox function 132

make-timer function 133
merge-ef-specs function 131
multi-column-list-panel class 137

N

null-pointer-p function 124

O

OLE control 106
OLE embedding 106
ole-control-component class 106
open function 114, 122
OpenGL 109
opengl-pane class 110
openssl-version function 114
open-tcp-stream function 114
option-pane class 110
output-pane class 136

P

packaging
  biodegradable 137
  recyclable 137
parse-float function 131
poor performance 99
print-base variable 131
process-arrest-reasons function 133
process-continue function 133
process-lock function 132
process-read-event function 132
process-run-function function 135
process-stop function 133
process-stopped-p function 133
process-unlock function 132
process-unstop function 133
process-wait-for-event function 132
profile macro 99
profiler-tree-from-function function 113
profiler-tree-to-function function 113
prompt-for-file function 108

Q

quit
  IDE behavior 116

R

read function 130
recording-for-undo macro 120
room function 99

S

Save modified buffers
  action item 127
save-image function 12, 106, 111, 132
schedule-named-event function 133
schedule-timer-relative function 133
screen-active-interface function 109
set-file-dates function 131
set-memory-check function 114
set-memory-exhausted-callback function 114
set-object-automatic-resize function 108
setup-for-alien-threads function 132
set-up-profiler function 113
set-verification-mode function 114
set-window-split-on-space function 109
simple-pane class 106
slot-definition class 128
slot-definition-allocation accessor 128
SQL operators
  LIKE 126
SQL pseudo operators
  sql-boolean-operator 126
  sql-operator 126
sql-boolean-operator SQL pseudo operator 126
sql-operation function 126
sql-operator SQL pseudo operator 126
standard-slot-definition class 128
start-environment function 11, 84
start-up-server function 114
stream-file-position generic function 131
subtypep function 127
switchable-layout class 108
Symbol Browser 118
symeval-in-process function 115

T

tab-layout class 108
terminal-io variable 135
text-input-pane class 109, 136
translate-logical-pathname function 129
tree-view class 107, 136, 137
tree-view-update-an-item function 110
tree-view-update-item function 110
typep function 127

V

variables
dont-undo 120
enter-debugger-directly 135
features 128
print-base 131
terminal-io 135

W

with-buffer-locked macro 120
with-dynamic-foreign-objects macro 124
with-dynamic-foreign-objects-with-cleanups macro 125
with-external-metafile macro 106
with-heavy-allocation macro 114
without-interrupts macro 120
without-preemption macro 120
with-output-to-fasl-file macro 129
with-page macro 136
with-point-locked macro 120
:**wrap-style** initarg 109