
Release Notes for Allegro CL 4.3 on various platforms

This document¹ describes Allegro CL 4.3. This document is needed by all users of Allegro CL 4.3 on Sun 4's and SPARCstations (and related machines), HP 9000/7xx series workstations, IBM RS/6000 workstations, Silicon Graphics workstations, and DEC Alphas.

1 Introduction

These Release Notes describe Allegro CL 4.3 compared to Allegro CL 4.2. This document is divided into the following sections:

1. **Introduction.** The section you are now reading.
2. **Release inventory.** This section lists the contents of the distribution package and the additional software supplied on the distribution CD.
3. **Documentation.** This section describes the documentation for Allegro CL 4.3.
4. **New and changed features.** This section describes the new and revised features in Allegro CL 4.3.
5. **Things to be aware of and known problems.** This section describes issues users should know about and known problems with the 4.3 release.
6. **Machine requirements and information.** Subsections give information pertinent to various platforms. Please read the section that applies to your platform.
7. **How to contact us.** This section contains the usual boilerplate on contacting us and reporting bugs. (A similar section appears in almost all Franz Inc. documents.)

2 Release inventory

With the Allegro CL 4.3 distribution, you should have received:

- A CD containing the software distribution.
- The *Allegro CL 4.3 User Guide*

1. This document is Copyright © 1996 by Franz Inc. All rights reserved. This is revision 2 of this document, dated April 19, 1996. It has Franz Inc. part number D-R-00-GEN-01-60419-0-2
Allegro CL and Allegro Composer are registered trademarks of Franz Inc.
Allegro Common Windows, Allegro Runtime, and Allegro Presto are trademarks of Franz inc.
Unix is a trademark of AT&T; SunOS and SPARCstation are trademarks of Sun Microsystems; HP and HP/UX are trademarks of Hewlett-Packard; IRIX and Iris are trademarks of Silicon Graphics; Motif is a trademark of OSF Inc. DEC Alpha is a trademark of Digital Equipment Corp.

-
- *Installation Guide for Allegro CL 4.3, Allegro Common Windows 2.1, Allegro Composer 2.0, CLIM 2.1 on various platforms* (which describes installation on all platforms just as these Release Notes are for all platforms)
 - *Release Notes for Allegro CL 4.3* (the document you are now reading)
 - *Technical Memorandum #23: Delivering Applications in Allegro CL.*

If you ordered the International Character Set version of Allegro CL, you will also get *Technical Memorandum #24: International Character Set Allegro CL*. There are also Release Notes for Allegro CLIM 2.1 and Allegro Composer, sent to customers who order those products. Other documentation for those products may be supplied (see the appropriate Release Notes for information).

Directories on the distribution CD

The following directories should be found on the distribution CD. Note that *build/* and *home/* are actually encoded subdirectories of directories named by platform (sunos4, dec, ibm, etc.). You are given a key which allows you to decode and copy the relevant directories for the products you have licensed.

build/

home/

These directories are used for building Allegro CL and related products. (Note that *home/* was called *lib/* in earlier releases.) The *examples/* subdirectories of *home/* has example code for Allegro CL and related products.

extra/gnu_src/

Sources for Emacs distributions and utilities.

extra/aclwin/

A free (but limited) version of Allegro CL 3.0 for Windows.

extra/

This directory contains various things that may be of interest in addition to the items listed above.

3 Documentation

The documentation for Allegro CL 4.3 consists of:

- These Release Notes
- The Installation Guide
- The *Allegro CL 4.3 User Guide*
- Technical Memorandum #23: Delivering applications in Allegro CL
- Technical Memorandum #24: International Character Set Allegro CL (ICS customers only).

4 New and changed features

Note for users of CLIM

Allegro CL 4.3 only supports one version of CLIM: CLIM 2.1 with Motif look-and-feel. CLIM 1.x and CLIM 2.1 with OpenLook look-and-feel are no longer supported. Please see the *CLIM 2.1 with Allegro CL 4.3 Release Notes* for more information on CLIM.

Non-backward compatible changes

The following changes in 4.3 (compared to 4.2) require code changes. Unchanged code will likely fail.

1. **dumplisp** arguments have changed.
 - `:restart-function` argument removed (set `*restart-init-function*` or `*restart-app-function*`, documented in chapter 8 of the *Allegro CL User Guide*, instead).
 - `:restart-actions` argument removed (set `*restart-actions*` instead -- note change in format, documented in chapter 8 of the *Allegro CL User Guide*).
 - `:read-init-file` argument removed (set `*read-init-files*` instead, documented in chapter 8 of the *Allegro CL User Guide*).
2. **install_lisp** arguments have changed. CLIM 1.1 and OpenLook versions no longer supported, so `clim=`, `libXol=`, `whichclim2`, `clim2ol=` arguments all removed.
 - `restart-function=` removed (set `*restart-init-function*` or `*restart-app-function*`, documented in chapter 8 of the *Allegro CL User Guide*, instead).
 - `prealloc=` has been replaced with the more accurately named `estimated_max_heap_size=`, but `prealloc=` is still accepted.
3. **make-static-array** removed. **make-array** now accepts `:allocation` keyword argument. `:allocation :static` means make a static array. See chapter 15.
4. `ALLEGRO_LIBRARY_PATH` (as a method of finding loaded `.so` files) no longer used. See the discussion in chapter 8 on using logical pathnames to load (and later find) `.so` files.
5. The home packages of certain symbols have changed. (The change allows certain packages, notably the multiprocessing package, to be left out when not needed -- the moved symbols are always needed). The following table shows the symbol, the 4.2 home package, and the new, 4.3 home package.

| Symbol | Old (4.2) home package | New (4.3) home package |
|----------------------------------|------------------------|------------------------|
| <code>*all-processes*</code> | multiprocessing | system |
| <code>*current-process*</code> | multiprocessing | system |
| <code>global-symbol-value</code> | multiprocessing | system |

6. The following symbols that related (in earlier releases) to the Lisp library have been removed from Allegro CL 4.3: `excl::*library-code-pathname*`; `excl::*library-pathname*`; `excl::*library-code-fasl-pathname*`; `excl::*library-code-cl-pathname*`; `excl::*library-doc-pathname*`; `excl::*library-code-bin-pathname*`. In 4.3 there is no variable that holds the Lisp home location (called the library in earlier versions). Evaluate the following to get the location:

```
(translate-logical-pathname "sys:")
```

7. The call-counting profiler has been integrated with the space and time profilers. All functions associated with the call-counting profiler have been removed. See chapter 11 of the *User Guide* for more information.

Other changes

What is new in 4.3 compared to 4.2? Here are some of the user-visible changes in 4.3.

1. **Finding files Lisp needs to run.** Lisp will now use an environment variable (if it is set) to find files it needs to run. If the variable is not set, Lisp tries the location stored in the image. (Contrast this with the 4.2 behavior where Lisp looked first in the location stored in the image and then in various locations around the directory containing the executable.) See the *4.3 User Guide* for detailed information on this issue. Images can be declared standalone, which means no files will be looked for on startup. Again, see the *User Guide* for more information.
2. **You can include .so files in image at build time [DLFCN platforms only].** If you specify a *.so* file on the `install_lisp` command line, that file will be linked into the image and available on startup just as if it had just been loaded. Such *.so* files are found when the image is started with `LD_LIBRARY_PATH`, just as similar files are with any UNIX program. The image will not start if the *.so* file cannot be found (again, as with other UNIX programs).
3. **No limit on number of registered, C-callable Lisp functions.** In 4.2, up to 100 functions could be registered with `ff:register-function` (see page 10-39) but no more than 100. In 4.3, there is no hard limit on the number of functions that can be registered. Please note that each registered function uses up to 40 bytes (depending on platform) in malloc space and a Lisp vector is allocated with as many elements as the largest registered function index. We do recommend that indices be reused when possible.
4. **Changing case mode.** Case mode should only be set when Lisp is built. Changing case mode in an already-built image is no longer supported, though it usually works except when the image is built with the `+presto` argument to `install_lisp`.
5. **Ghost frames are printed with square brackets in brief backtraces.** In 4.2, they were indicated with square brackets in moderate and verbose backtraces and with parentheses in brief backtraces. See section 5.3.8 **Ghost frames in backtraces** in the *4.3 User Guide*.
6. **Random improved but only for certain arguments.** In earlier releases, the version of `random` supplied with Allegro CL was slow and inefficient (but random enough). Users could get a better version from Franz Inc. upon request. In 4.3, the better version is integrated into the product, but is only used for arguments `1.0s0` (i.e. single-float 1.0) and fixnums. All other arguments cause `random` to use the old, inefficient generator. See the discussion in chapter 3 of the *User Guide* for more information.
7. **Lisp startup code source available.** The file `home/code/aclstart.cl` is the source for the startup of Allegro CL. Programmers are encouraged to examine it to understand the startup procedure in detail and to figure out how to affect startup as they might wish. The procedure is documented in chapter 2 of the *User Guide*.
8. **Application packaging program supplied.** `home/code/launcher.cl` is the source for a program that, from within a running Lisp, dumps an image and copies all necessary files into a specified directory and writes a shell script to be used in association with the application. See the code and Technical Memorandum #23 for more information.

5 Things to be aware of and known problems

In this section, we first mention a number of things that you should be aware of. Then we describe known problems with Allegro CL 4.3.

Things to be aware of

There are a number of issues that users should know about. Some items apply to any release and are unchanged since the 4.2 Release Notes. Those items are indicated with **[Unchanged from 4.2]**.

-
1. **You must recompile all files.** *fasl* (compiled Lisp) files created by any version of Allegro CL earlier than 4.3 final will not load into Allegro CL 4.3. Please recompile all source files.
 2. **[Unchanged from 4.2] Reading from /dev/null on Solaris 2.x.** A Lisp image started with `-batch` option reads input from `stdin` and exits when an EOF is encountered (or when an error occurs). On Solaris 1.x, starting Lisp (`cl` names the Lisp image) as follows would print 'Read this form!' and exit, as you would expect:

```
% cl -batch -e '(format t "Read this form!~%")' < /dev/null
```

On Solaris 2.x, this does not work. `poll`, the system command that accesses `/dev/null` does not signal an EOF. Instead, it waits for input and thus Lisp waits, as if hung. This may be a bug in Solaris 2.x, or it may be intended behavior. In any case, Lisp run in batch mode does not work as expected. If you used this idiom, put an explicit call to `excl:exit` in the `-e` form, as follows, and you will get equivalent behavior:

```
% cl -batch -e '(progn (format t "Read this form!~%") (excl:exit))' < /dev/null
```

3. **[Unchanged from 4.2] There is a space cost to source file and cross-reference recording.** Allegro CL permits recording of information on the source of function definitions and on cross-reference within a *fasl* file. Users should be aware that there is a space cost to storing this information. A *fasl* file created with `excl:*record-source-file-info* t` is typically 5 to 10% larger than the file created with that variable `nil`. Similar percentages apply to *fasl* files created with `excl:*record-xref-info* t`. If both are true, the files can, therefore, be 10 to 20% larger. If the files are loaded with `excl:*load-source-file-info* and excl:*load-xref-info* true`, the additional space is added to the image. Users should balance the usefulness of the information with the increased space cost.
4. **[Unchanged from 4.2] svref cannot be applied to CLOS instances and Flavor instances.** In earlier releases, some users were advised that they could apply `svref` to CLOS instances and flavor instances for certain optimizations but this has not been possible since version 4.2.
5. **change-class.** It now takes `initargs` that are passed to `update-instance-for-different-class`.
6. **ignore declaration and unused variables.** In 4.2, setting a variable sufficed for it to be used, so if a variable was set (regardless of whether it was referenced), no 'variable never used' warning would be signaled. As described in Appendix A, item 359, this was at variance with the ANSI standard. 4.3 now conforms to ANSI in this regard, so compiling the following function in 4.3 will generate a warning (it did not in 4.2):

```
user(1): (defun foo ()
          (let ((x 10) (y 5))
              (setq y (+ 1 x))))
foo
user(2): (compile 'foo)
; While compiling foo:
Warning: Variable y is never used.
foo
t
nil
```

Known problems

1. **[Unchanged from 4.2] The time profiler and interrupted sampling.** As described in chapter 11 of the *Allegro CL User Guide*, you can provide fine-grained control over sampling with the time (and space) profiler with the functions `prof:start-sampling` and `prof:stop-sampling`. These functions start and stop collection of data during a profile run. Unfortunately, the total time calculated for the profile is always the machine time from the beginning to the end

of the profile -- that is it includes the time when sampling is stopped. Since times are expressed as a fraction of the total time, all times will be inflated when fine-grained control is used. However, the fraction of time spent in each function (as opposed to the actual time) will be correct (except for statistical variation). We have left this bug in the system since fixing it is complicated and we believe that it is the fraction rather than the total time that is of interest when using fine-grained profiling.

2. **[Unchanged from 4.2] compile in a compilerless Lisp only returns one value.** The function `compile` should return 3 values -- as documented in, e.g. *Common Lisp: the Language* (2nd ed.) However, Allegro CL provides the option of installing a Lisp image without the compiler and in such an image, the function `compile` (which exists but does not, of course, do much) returns only one value. Users who will be using compilerless images should use care not to expect 3 values returned from `compile` when they call that function. We assume that most users will use images with the compiler and thus will be wholly unaffected by this bug.

6 Machine requirements and information

- **SPARCstations and Sun 4's:** see section 6.1. Users running Solaris 2.x, see also section 6.1.1. Users running SunOS 4.1.x, see also section 6.1.2.
- **HP 9000/7xx workstations:** see section 6.2.
- **Silicon Graphics workstations:** see section 6.3.
- **IBM RS/6000 workstations:** see section 6.4.
- **DEC Alpha workstations:** see section 6.5.

6.1 Information for users on SPARCstations and Sun 4's

There are two versions of Allegro CL 4.3. One that runs on Sun 4's and SPARCstations running the Solaris 2.x/SunOS 5.x operating system. The other runs on the Solaris 1.x/SunOS 4.1.x operating system. Note that these two versions are not compatible -- the Solaris 1.x image will not run on a Solaris 2.x machine and vice versa. *fasl* (compiled Lisp) files, however, are compatible between versions.

The Solaris 1 version will not work with SunOS 4.0. You must have SunOS 4.1 or later installed.

The proper version of Allegro CL for the Operating System you are running will work on any Sun 4 or SPARCstation.

The same distribution that runs on Sun 4's and SPARCstations will also run on Solbourne SPARC-based computers (Solbourne computers typically run an Operating System compatible with Solaris1.x/SunOS 4.1.x.). Solbourne users should note the following (which does not affect users of Sun computers):

The X server on Solbourne machines may exhibit a bug with its backing store. This bug causes some Allegro Composer windows to come up partially or wholly blank. If this happens, users should evaluate the following form to turn off backing store when using Allegro Composer with such a server:

```
(setq cw:*default-use-backing-store-p* nil)
```

6.1.1 Important information for Solaris 2.x/SunOS 5.x users

OS version

You must be running Solaris 2.2/SunOS 5.2 or later. The latest OS version at the time this document was written was SunOS 5.5. Please contact Franz Inc. if you have problems with Allegro CL or related products running on a version of SunOS later than 5.5.

With Solaris 2.x, various Sun tools must be installed

Users running Solaris 1.x should skip to section 2.1.2. If you are running Solaris 2.2 or later, the following SunOS packages must be installed:

SUNWlibm SPARCompilers Bundled libm
SUNWhea Header Files
SUNWbtool CCS tools bundled with SunOS
SUNWarc Archive Libraries
SUNWtoo Programming Tools
SUNWsprout SPARCompilers Bundled tools

If you intend to use CLIM, the following must be installed on the machine where the image is built and on all machines where the image is run:

SUNWmfrun Motif RunTime Kit

If these tools are not installed, the installation will inform you (and not complete). Also, the Sun utility **pkginfo**, typically in `/usr/bin/`, can be used to list installed packages. Please contact your Sun representative for information on obtaining these tools if you do not already have them.

Foreign loading on Suns running Solaris 2.x

Allegro CL on Suns running Solaris 2.x uses the **dlfcn** model for foreign loading. `:dlfcn` appears on `*features*`. `.o` files should be generated with `-K pic`; `.so` files with `ld` and the argument `-G`. The information on creating `.so` files was correct when this document was written, but may change because of actions of the hardware vendor. Please see the vendor-supplied documentation for complete information on creation of shared object files. Also note the following:

On Solaris 2.x, your PATH environment variable should have `/usr/ucb` after the directory containing your C and Fortran compilers and after `/usr/ccs/bin`. Foreign (compiled C or Fortran) code must be compiled with a compiler that generates position independent code. The `/usr/ucb/cc` C compiler is known not to produce such code. Therefore, the directory containing a position independent C compiler must precede `/usr/ucb` in your PATH. (The directory containing the C compiler differs from system to system. It is often something like `/opt/SUNWspro/bin`.) Similarly for the Fortran compiler. `/usr/ucb/ld` also does not work correctly with Allegro CL. The directory containing the correct `ld`, typically `/usr/ccs/bin`, should precede `/usr/ucb`.

The LD_LIBRARY_PATH environment variable must be set correctly.

This variable is used by `dlopen()` to determine the locations of libraries needed to run Lisp and needed by foreign files loaded into Lisp. Please check with your system administrator or refer to your Sun documentation for correct settings of this variable. If you use CLIM, this variable must include a reference to the location of the Motif library, as described under the large heading **Note for users of CLIM of Solaris 2.x** below.

Notes for users of CLIM on Solaris 2.x

- Allegro CL 4.3 only supports one version of CLIM: CLIM 2.1 with Motif look-and-feel. CLIM 1.x and OpenLook look-and-feel are no longer supported.
- `/usr/dt/lib` must be included in `LD_LIBRARY_PATH` (by the user who builds a CLIM image and by all users who run the CLIM image.)
- The Sun tool **SUNWmfrun** must be installed on the machine where a CLIM image is built and on all machines where the image is run (see the beginning of this section).
- Allegro CL 4.3 for Solaris 2.x uses the Sun-supplied version of Motif. (In earlier releases, Motif was included with the Allegro CL distribution.)

6.1.2 Important information for Solaris 1.x/SunOS 4.1.x users

Allegro CL 4.3 will not run on machines running SunOS 4.0. You must have SunOS 4.1 or later for Allegro CL 4.3 to work.

A version of Allegro 4.3 for Solaris 1.x/SunOS4.1.x will not run on a machine running Solaris 2.x/SunOS 5.x.

Foreign loading on Solaris 1.x

Allegro CL on Suns running Solaris 1.x uses the **non-dlfcn** model for foreign loading. `:dlfcn` is not on the `*features*` list. See chapter 10 of the *Allegro CL User Guide* for more information on foreign functions and the `dlfcn` and `non-dlfcn` models for foreign loading.

Notes for users of CLIM on Solaris 1.x

- Allegro CL 4.3 only supports one version of CLIM: CLIM 2.1 with Motif look-and-feel. CLIM 1.x and OpenLook look-and-feel are no longer supported.
- Allegro 4.3 for Solaris 1.x comes with all necessary Motif functionality included. It is not necessary to purchase or install Motif to use CLIM.

6.2 Information for users on HP 9000/7xx machines

This section should be scanned by everyone installing Allegro CL. New users may want to read it more carefully. The topics covered are under descriptive headings in large, **bold type**. See below for information about foreign loading on HP machines.

You must be running HP-UX 9.05 or later

Allegro CL 4.3 is not supported under versions of HP-UX earlier than 9.05. Please upgrade to that release of HP-UX if you have not already done so. If you have problems running under a later version of HP-UX, please contact us for assistance. Information on contacting us can be found at the end of this document.

Note that under HP-UX 10.0, you cannot build a CLIM image (because the necessary libraries exist in shared form only -- there are no static equivalents). The pre-built CLIM image supplied with the distribution works on HP-UX 10.0 as should any image built under HP-UX 9.05.

/usr/lib/libBSD.a must be installed

To install or use Allegro CL, you must have `/usr/lib/libBSD.a` installed on your machine. Please contact your HP representative for assistance if this library is not present on your machine.

Developers Toolkit fileset required for development in CLIM 2

The Developers Toolkit fileset must be installed in order to develop programs in CLIM 2 (because it contains necessary Motif libraries). Please contact your HP representative if you need assistance obtaining the Developers Toolkit fileset. Note that the pre-built CLIM 2 binary supplied with the Allegro CL distribution (to licensed CLIM users) has all necessary functionality already included so you can work with that image without having the Developers Toolkit fileset installed.

Foreign loading and image type on HP workstations

- Allegro CL on HP's uses the **non-dlfcn** model for foreign loading. `:dlfcn` is not on the `*features*` list. See chapter 10 of the *Allegro CL User Guide* for more information on foreign functions and the `dlfcn` and `non-dlfcn` models for foreign loading.
- **Allegro CL images are EXEC_MAGIC objects** (in earlier releases they were SHARED_MAGIC objects). Don't worry if you do not know the difference between EXEC_MAGIC and SHARED_MAGIC; what is important is the effect of being a EXEC_MAGIC object. In EXEC_MAGIC, the Lisp heap can be 1.9 Gbytes large (instead of the previous .9 Gbytes).
- **Foreign loading and excl:dumplisp.** If you use `excl:dumplisp` to dump an image in which foreign files have been loaded, the restarted Lisp will not know the locations of the loaded foreign files. All `defforeigns` (definitions of Lisp functions connected to foreign code) will be valid in the dumped image but new `defforeigns` cannot be done on foreign code loaded before the image was dumped. (This restriction does not apply to `.o` and `.a` files included in the image when it is installed with `build/install_lisp`. `.o` and `.a` files can be put on the `build/install_lisp` command line for inclusion in the image. See the Installation guide for more information.)
- **Apparent bug in ld may cause foreign code failure.** There is an apparent bug in `ld` where certain global data is not properly linked. Specifically, if a file defines a function, and another file uses the address of that function, then the dynamically loaded code may get the address of the wrong function. If you have a problem with foreign code, try linking in that code when you install the image, to see if the error goes away. (You link foreign code when you build the image by

putting the necessary *.o* and *.a* files on the `build/install_lisp` command line. See the Installation Guide for details.)

For example, the X11R5 library (*libX11.a*) exhibits this problem, so if you plan to use this library, it is necessary to include it with the Lisp when you build the image. (If you build an image with Allegro CLIM, this is done automatically.)

6.3 Information for users on Silicon Graphics workstations

OS version

You must be running IRIX 5.3 or later. If you are running an earlier version of IRIX, please upgrade. If you are running a later version and experience problems, please contact us for assistance. Information on contacting us can be found at the end of this document.

You must have the right processor chip

This version of Allegro CL will work on all machines with

- an R3000 processor chip
- an R4xxx processor chip where at least one 'x' is not 0 -- i.e. R4600
- an R4000 Rev. 3.0 or later processor chip

Allegro CL will *not* work on machines with Rev 2.x (or 1.x if it exists) R4000 processor chips. The `hinv(1)` command can be used to determine the chip type and rev number.

Foreign loading on SGI workstations

Allegro CL on SGI's uses the `dlfcn` model for foreign loading. `:dlfcn` appears on `*features*.so` files should be generated with `-KPIC`; `.so` files with `ld` and the argument `-shared -all`. Note the `-all` causes all entry points to be included. If a library `libx.a` or `libx.so` is included as part of the shared-object creation, the `-all` option may have to be nullified by the inclusion of a `-none` option just before the libraries are included. The information on creating `.so` files was correct when this document was written, but may change because of actions of the hardware vendor. Please see the vendor-supplied documentation for complete information on creation of shared object files.

6.4 Information for users on IBM RS/6000 workstations

After the various topics in this section, a subsection discusses foreign loading on the RS/6000 machines in detail.

You must be running AIX 3.2.5 or 4.1

Allegro CL 4.3 works on AIX 3.2.5 and AIX 4.1. Please upgrade to one of those releases of AIX if you have not already done so. Please contact us if you have a later version and experience problems. Information on contacting us can be found at the end of this document.

Necessary IBM-supplied software and setup

- To build the Lisp image on the RS/6000, certain (optional) IBM-supplied modules must be available. To build a vanilla Lisp image, you must have `xlccmp.obj`, all the `bos.ext1` modules, and all the `bosadt` modules

To use the Emacs-Lisp interface, you must have in addition the `bosnet.tcpip.obj` and the `bosnet.nfs.obj` modules. To use CLX, Allegro Common Windows, and Allegro Composer, one must have in addition *all* of the X11 modules.

You can examine the modules available on your machine with the `lslpp` system command. Thus the following command will display the modules available:

```
% lslpp -h | more
```

- Lisp images can be quite large. The system parameters on the RS/6000 specify a maximum file size per user. That maximum must be large enough for the images. We recommend setting the limit so files as large as at least 20 Megabytes are permitted. A larger value may be necessary for some applications. The maximum must be set before Lisp is installed or the installation process may fail.

The maximum can be set using the SMIT system utility. Choose `Security and Users` from the first menu, `Users` from the next, and `Change/Show characteristics of a user` from the next. Enter the name of the user and find `Max File Size` and set it appropriately. (Warning: we are describing IBM system software over which we have no control. These instructions are correct as of the date of this document but IBM may have made changes since that time. Please consult IBM documentation on SMIT for complete and up to date information.)

- The amount of swap space specified in the default configuration of your workstation may be inadequate to run Lisp. We recommend a minimum of 30 Megabytes of swap - more if more than one user will be using Lisp or if your application is very large. Lisp images are typically 5 to 20 Megabytes. Allegro Common Windows and Allegro Composer each add an additional 2 megabytes. You may need to reconfigure your machine to increase swap space. Please refer to your system administration manuals for details on reconfiguring your machine.

A kernel extension for AIX is provided with the distribution

This extension improves foreign loading. It does not otherwise affect AIX or your environment. The code and instructions for installing it are in the `kernel_extension` directory read off the distribution CD. See the Installation Guide for information on installing the kernel extension.

If it is not possible to install the kernel extension (for whatever reason), foreign loading will in fact work. However, performance is degraded and certain coding restrictions apply. See the information under the heading **Foreign loading without the kernel extension** in 6.4.1 **Loading foreign code on IBM RS/6000 workstations** below for more information. Nothing in the installation procedure depends on the kernel extension being installed.

6.4.1 Loading foreign code on IBM RS/6000 workstations

Like release 4.2.1 (but unlike releases previous to that) Allegro CL 4.3 on the RS/6000 *does* support dynamic loading of foreign files into a running Lisp image. Loading follows the DLFCN model described in section 10.1.1 of the 4.3 *User Guide* **The DLFCN model for loading foreign code**. You can also include foreign code with the image when it is built. See section 6.1 **Including foreign code in an Allegro CL image** in the Installation Guide for more information.

In the remainder of this section, we describe loading of foreign code into Allegro CL 4.3 on IBM RS/6000 workstations. An IBM-supplied kernel extension which makes foreign loading more efficient is included with the Allegro CL 4.3 distribution. You are strongly urged to install that kernel extension, but if you cannot for any reason, be sure to read the information under the heading **Foreign loading without the kernel extension** below.

Use section 10.1.1 of the 4.3 User Guide for loading code

- The PICFLAGS variable (that specifies the correct arguments to **cc** for position-independent-code) should be the empty string or simply undefined. All files compiled on the RS/6000 are already position-independent so no new flags are needed. Therefore, source files should be compiled as follows:

```
cc -c foo.c
```

- Shared object files are created with **make_shared**, not with **ld**. Like PICFLAGS, SHAREFLAGS should be the empty string or simply undefined. Create a *.so* file with the following command:

```
make_shared -o foo.so foo.o [more .o files] [-l<lib1> -l<lib2> ...] [exp-files]
```

(*exp-files* are explained under the next heading.) The **make_shared** program is included in the Allegro CL 4.3 distribution and is placed in the same directory as the Allegro CL image. That directory *must* be in your PATH environment variable (as it typically will be since the directory containing the Allegro CL image should also be in your PATH). Note that **make_shared** calls the program **make_exp**, which is also placed in the Allegro CL image directory.

Exports (.exp) files

The AIX operating system does not make any symbols available for dynamic loading except for those symbols that are specifically exported by the **make-shared** command. This is in direct contrast to other operating systems, which tend to look at any external symbols as candidates for linking at load time. The method for identifying these exported symbols is via an *exports* file.

Whenever a Lisp image is built, a file is created with the same name as the Lisp image and with extension *.exp*. The *.exp* file is stored in the same directory as the Lisp image. Similarly whenever a *.so* file is built, a *.exp* file is created that corresponds to the *.so* filename. These files are exports files, and are available for use, along with AIX-supplied *.exp* files, to import symbols during the building of the *.so* file. To import symbols from a *.exp* file, simply include the *filename.exp* on the **make_shared** command line.

```
cc -c foo.c
make_shared -o foo.so foo.o /usr/local/bin/lisp.exp -lm
```

Other facts

- **dlopen()**, **dlsym()**, **dlerror()**, and **dlclose()** for the most part operate in the same way as they do for SVR4 operating systems. The one notable exception is that a NULL handle to **dlsym** (i.e. **dlsym(NULL, string)**) is not defined as it is in some SVR4 versions.

-
- Allegro CL 4.3 for the RS/6000 includes `:dlfcn` on the `*features*` list, thus allowing `#+dlfcn` and `#-dlfcn` to be used for conditional compilation. The RS/6000 version of Allegro CL does not have `:svr4`, however, since it not an SVR4 machine.

Dumplisps will be somewhat bigger than necessary

Certain information about foreign code loaded into a Lisp is lost when you dump an image with `excl:dumplisp`. This loss is not user-visible (the system will reconstruct the information when the dumped image is restarted) but the space where the information is stored is not recovered. That space is wasted in the dumped image, and if you dump the dumped image, more space will be wasted. (The amount of wasted space is approximately the size of the loaded `.so` files.) The amount is not large, but many repeated `dumplisps` are not recommended. Remember that you can include foreign code directly in the Lisp image when it is built (as described in the Installation Guide). It is often useful to do so when you know that foreign code will be needed.

Foreign loading without the kernel extension

Included in the Allegro CL 4.3 distribution is an IBM-supplied kernel extension to AIX. We strongly recommend that you install this extension. (The code is in the `kernel_extension` directory on the distribution CD. The `README` file in that directory describes the extension and tells how to install it.) However, if you are unable (for whatever reason) to install the extension, foreign loading will still work as described here. It will, however, be less efficient, both in performance and in space usage. (If you look at `room` output after a foreign load, you will see more gaps. Foreign loading and foreign calls will both be slower.) Also, the (undocumented) function `ff:get-extern-data-address` will not work correctly.

Standalone images need to find routines for foreign loading

Standalone images (created with the `:standalone` argument to `excl:dumplisp` specified true as described in section 8.3 of the *User Guide*) do not set up a translation for the `sys:` logical host. The program `dlsym_wrapper`, external to Lisp, is needed to load foreign code on the RS/6000. This program is looked for in the Lisp home location (the translation of `sys:`) in a standard image. In a standalone image, it is looked for in the directory where the Lisp image lives. Thus the image is not truly standalone and programmers must remember to ship `dlsym_wrapper` with the image if dynamic foreign loading is desired.

6.5 Information for users on DEC Alpha workstations

You must be running Digital Unix 3.2

Allegro CL 4.3 has only been tested on Digital Unix (formerly DEC OSF-1) 3.2. Please contact us if you experience problems with a later version.

Pointer size in Allegro CL on the DEC Alpha

The DEC Alpha is a 64-bit machine but it also provides support for 32 bit pointers. The foreign function interface assumes the standard 64 bit pointer size. This allows standard 64 bit libraries and user foreign code to be linked or loaded without special treatment. Note however that on the Alpha `ints` and `longs` are not the same size and that pointers are not the same size as `ints`.

Internally Allegro CL uses 32 bit pointers in Lisp objects. If user foreign code includes `lisp.h` to examine and manipulate Lisp structure directly it should be compiled using the various `-tas0` options to allow for 32 bit pointers. Details on using these options can be found in the document "DEC OSF/1 Programmer's Guide Appendix A, Using 32 bit Pointers on DEC OSF/1 systems."

Foreign loading on DEC Alpha workstations

Allegro CL on DEC workstations uses the `dlfcn` model for foreign loading. `:dlfcn` appears on `*features*`. The C compiler always produces position independent code so no special arguments are required for `cc` (i.e. the `PICFLAGS` variable in section 10.1.1 should be the empty string). `.so` files are created with `make_shared`, supplied with Allegro CL, as described next.

Shared object files are created with `make_shared`, not with `ld`. Create a `.so` file with the following command:

```
make_shared -o foo.so foo.o [more .o files] [-l<lib1> -l<lib2> ...]
```

The `make_shared` program is included in the Allegro CL 4.3 distribution and is placed in the same directory as the Allegro CL image. That directory *must* be in your `PATH` environment variable (as it typically will be since the directory containing the Allegro CL image should also be in your `PATH`).

See chapter 10 of the *Allegro CL User Guide* for more information on foreign loading and foreign functions.

7 How to contact us

Please contact us if you have any problems with or questions about Allegro CL. Our voice phone number is (510) 548-3600. We are open from 8:00 am to 5:00 pm (Pacific time) every business day and technical people are available during those times to assist you. Our FAX number is (510) 548-8253. You may send us a fax at any time.

The best way to send us a bug report is by electronic mail (e-mail). E-mail allows you to send exact transcripts of Lisp sessions. You can send us e-mail at the following address:

bugs@franz.com

In any message, be sure to include your name and organization, your e-mail address, your phone number (very important), the machine make and type, and the operating system version identifier. Also include the Allegro CL version number, and as much information as you can give us about the problem, including, if possible, a small test case illustrating the problem. We recommend that Lisp sessions be recorded with the **excl:dribble-bug** function when preparing a bug report or question. This function, described in section 1.7 **Reporting bugs** in the *Allegro CL User Guide*, is similar to the Lisp function **dribble** with the added feature that it automatically records information about your version of Allegro CL in the dribble file.

We need you to explicitly tell us your e-mail address in the body of the letter since the return addresses generated by mailers may be invalid. Also we need your phone number in case e-mail to you fails.

We are also interested in your comments about our product. There is an information sheet in the *Allegro CL User Guide* with more electronic addresses for comments. You may also use one of the addresses above.

The mailing list, `Allegro-CL@cs.berkeley.EDU`, is used by Franz Inc. for announcements related to the Allegro CL family of products. Customers can use it to start discussions with other customers. This list is unmoderated and completely open. If you want your name added to the mailing list, send a request to `Allegro-CL-request@cs.berkeley.EDU`.

Note that any technical communication with Franz Inc. should be sent to `bugs@franz.com` (i.e. the address near the top of the page). Despite the name of that mailing list, we welcome any question or comment, not just bug reports.

We hope you enjoy our product. We appreciate having you as a customer.

New WWW page

Please check out the Franz Inc. World Wide Web page, <http://www.franz.com/>. It contains information about Franz Inc. products and allows easy access to patches and the various Franz Inc. product FAQ's.