

Release Notes for Allegro CL 8.0

This document contains the following sections:

[1.0 Introduction](#)

[2.0 Information on changes and new stuff since the 8.0 release](#)

[3.0 Fasl files are not-compatible between versions and operating systems](#)

[3.1 All pre-8.0 Lisp compiled files must be recompiled \(old fasl files will not load\)](#)

[3.2 Fasl files may be incompatible in different Lisps on same hardware](#)

[4.0 Release Notes for installation](#)

[5.0 Release notes for specific platforms](#)

[5.1 OS patch needed for Solaris 2.8 on Sparcs](#)

[5.2 HP Alpha running Tru64: default stack size should be increased](#)

[5.3 Notes of increasing the default maximum stack size on HP-UX 11 machines](#)

[5.4 Mac OS X notes](#)

[5.5 Heap start locations](#)

[6.0 Release Notes for the base Lisp](#)

[6.1 New features in the base Lisp](#)

[6.1.1 Features added to Allegro CL 7.0 after the initial release of Allegro CL 7.0](#)

[6.1.2 Major new features in Allegro CL 8.0](#)

[6.1.3 Other new features in Allegro CL 8.0](#)

[6.1.4 Multiprocessing and OS threads in Allegro CL 8.0](#)

[6.2 Non-backward-compatible changes in the base Lisp](#)

[6.3 Other changes to and notes about the base Lisp](#)

[6.4 Base Lisp platform-specific information](#)

[7.0 Release Notes for CLIM](#)

[8.0 Release Notes for Common Graphics and the IDE](#)

[8.1 Significant changes in Common Graphics](#)

[8.2 Non-backward-compatible changes in Common Graphics](#)

[8.3 Functionality to handle differences between Windows and GTK](#)

[8.4 Other changes in Common Graphics](#)

[8.5 IDE release notes](#)

[8.5.1 Opening projects from releases prior to 8.0](#)

[9.0 AllegroStore not available in Allegro CL 8.0](#)

[10.0 Release notes for SOAP](#)

[11.0 Release notes for Allegro RPC](#)

[12.0 Release notes for jLinker](#)

[13.0 Release notes for AllegroServe](#)

[14.0 Release notes for IMAP and SMTP, XMLutils](#)

[15.0 Release notes for The Emacs/Lisp interface](#)

[16.0 Documentation modifications in Allegro CL 8.0](#)

[17.0 Availability of CLX for Allegro CL](#)

[18.0 Release notes for Orblink](#)

[Appendix A. Tightening of ANSI Conformance in Allegro CL](#)

1.0 Introduction

This document provides the release notes for release 8.0 of Allegro Common Lisp and related products. Many sections are divided into non-backward-compatible changes (that produce different behavior in release 8.0 compared to release 7.0) and changes unrelated to backward-compatibility. Note that a bug fix is *not* considered a backward-incompatible change even if it does result in changed behavior because the previous behavior was erroneous.

You may wish to look at the 7.0 Release Notes, included in this distribution as the file *version-7.0-release-notes.htm*.

This document describes the changes, major and minor, from 7.0 to 8.0. Please look particularly at these sections:

- [Section 6.1 New features in the base Lisp](#): this section describes new features.
- [Section 6.2 Non-backward-compatible changes in the base Lisp](#): this very important section describes changes to Allegro CL which are **not** backward compatible. You may have to modify your source code in light of these changes.
- [Section 8.2 Non-backward-compatible changes in Common Graphics](#): [Windows customers only] this section describes changes to Common Graphics which are **not** backward compatible. You may have to modify your source code in light of these changes.
- [Section 9.0 AllegroStore not available in Allegro CL 8.0](#): the AllegroStore database product is not available in release 8.0. Instead, AllegroCache is provided as a database product.
- [Section 2.0 Information on changes and new stuff since the 8.0 release](#): this section currently has no content. It is a placeholder for descriptions of material added after the initial Allegro CL 8.0 release, along with other post-7.0 changes. Content will only be added to the section in documentation updates after the release of Allegro CL 8.0.

If you notice changed or unexpected behavior with an operator, variable, class, or facility, search for its

name in this document to see whether there is an entry concerning it.

2.0 Information on changes and new stuff since the 8.0 release

This section is designed for updates to this document. When new features are added after the Allegro CL 8.0 release, they will be discussed here.

3.0 Fasl files are not-compatible between versions and operating systems

3.1 All pre-8.0 Lisp compiled files must be recompiled (old *fasl* files will not load)

fasl files (compiled Lisp files) created by releases of Allegro CL prior to 8.0 (including 8.0.beta) will not load into Allegro CL 8.0. All such files must be recreated by compiling the associated Lisp source files. An error will be signaled if Lisp attempts to load an older, incompatible *fasl* file.

3.2 Fasl files may be incompatible in different Lisps on same hardware

Fasl files created on Windows x86 cannot be loaded into Linux or FreeBSD x86 Lisps

The Windows and UNIX operating systems are too different. However, FASL files (for the same Lisp version) can generally be shared between different UNIX operating systems on the same hardware. The general principles are as follows. Note that incompatibility may creep in for reasons outside our control. If *fasl* files are incompatible, recompile on the target machine.

Fasl files will usually be compatible between platforms when:

- the architecture (hardware) is the same
- the calling convention is the same (that is, the method of calling functions and the registers they use for arguments are the same)
- the method for generating signals can be made compatible (this cannot be done for Windows and UNIX, so they are incompatible for this reason)

It is up to the user to ensure that there are no os-specific dependencies, or that features (i.e. #+/#-) did not cause essential code to be excluded or extraneous code introduced that would cause a problem on the crossed architecture.

4.0 Release Notes for installation

1. **Version 8.0 uses the 7.0 installation procedure:** installation is described in *installation.htm*.
 2. **The distribution includes 8 bit and 16 bit character images** (this information is repeated from earlier Release Notes). Allegro CL has images that support 8 bit characters only, or 16 bit characters only. It is our expectation that most users will use the 16 bit images. Executables supplied with the distribution either have or do not have `8' in the name. Those with 8 in the name (mlisp8 and alisp8, e.g.) support 8 bit characters. Those without a number in the name support 16 bit characters. Image (dxl) files also come in 8 and 16 varieties. Again, 8 in the name means 8 bit character support. Character support for images and executables must match. Trying to start an executable with the wrong type of image fails.
 3. **No prebuilt Allegro Composer images in the distribution** (this information is repeated from earlier Release Notes). To create an Allegro Composer image, start Allegro CL and load *buildcomposer.cl*. That will produce *composer* and *composer.dxl*, or *composer8* and *composer8.dxl*. Allegro Composer is available on Unix only.
 4. **No prebuilt Allegro CLIM images in the distribution** (this information is repeated from earlier Release Notes). To create a CLIM image, start Allegro CL and load *buildclim.cl*. That will produce *clim* and *clim.dxl*, or *clim8* and *clim8.dxl*.
-
-

5.0 Release notes for specific platforms

Allegro CL 8.0 is known to work with the following minimal operating system versions. Allegro CL 8.0 runs on all operating system versions released (not in beta or pre-release form) as of December 1, 2005. See below for places to obtain information on operating systems released after that date.

Note that Allegro CL 8.0 **does not run on Windows 98/Me or on Mac OS X 10.3.**

32-bit platforms

- HP Tru64 UNIX 5.1 -- see [Section 5.2 HP Alpha running Tru64: default stack size should be increased](#)
- HP-UX 11.00 -- see [Section 5.3 Notes of increasing the default maximum stack size on HP-UX 11 machines](#)
- FreeBSD 4.x and 5.x
- Linux (x86), glibc 2.2
- Linux (PPC), glibc 2.3
- Apple Mac OS X 10.4 -- see [Section 5.4 Mac OS X notes](#)
- Microsoft Windows 2000/XP/Server 2003
- IBM AIX 5.1
- Sun Solaris 2.8 -- see [Section 5.1 OS patch needed for Solaris 2.8 on Sparcs](#)

64-bit platforms

- HP Tru64 UNIX 5.1 -- see [Section 5.2 HP Alpha running Tru64: default stack size should be increased](#)
- HP-UX 11.00 -- see [Section 5.3 Notes of increasing the default maximum stack size on HP-UX 11 machines](#)
- Apple Mac OS X 10.4 -- see [Section 5.4 Mac OS X notes](#)
- Linux (AMD64), glibc 2.3
- IBM AIX 5.1
- Sun Solaris 2.8 on Sparc
- Sun Solaris 2.10 on AMD 64

Franz Inc. cannot maintain the same release schedule as the many operating system providers on the many platforms we support. Usually Allegro Common Lisp and all of its component parts will work on new Operating System versions that become available after release. But sometimes Allegro CL patches or operating-system patches, or installation tweaks, will be required before Lisp will run on an updated system.

Franz Inc. usually finds out about any such issues soon after new operating system releases appear and devises any necessary patches or compatibility procedures. We maintain current information about operating system versions on this web page: <http://www.franz.com/support/osinfo.lhtml>. We strongly advise you to check that page before updating your operating system. If we know that the new operating system is compatible, or is compatible with certain patches, you will find the information there. Similarly if it is known to be incompatible. If the new operating system is not yet listed it may be that it

has not yet been tested.

5.1 OS patch needed for Solaris 2.8 on Sparcs

A problem with Solaris 2.8 where calls to **run-shell-command** can cause Lisp to hang is fixed by operating system patch 108993-18 (which supersedes the earlier patch 108827-36 for this problem). On Solaris 8, it can also be fixed by adding **/usr/lib/lwp** to the front of your `LD_LIBRARY_PATH` environment variable. No fix is necessary for Solaris 2.9.

The patch can be obtained from this Sun website: <http://sunsolve.sun.com/search/document.do?assetkey=1-26-49843-1>.

5.2 HP Alpha running Tru64: default stack size should be increased

The default stack size limit on an Alpha is 2 Megabytes, which is too low for normal stack overflow handling. Doing

```
limit stacksize unlimited
```

in a csh shell seems to allow up to 16 or 32 Mb, and users can run that command before running Allegro CL. (That command could be put into a `.cshrc` file.)

However, that only solves the problem for Allegro CL processes that are started from within that shell. You can also change the default for the machine as a whole by doing something like the following. (This procedure is provided as an example of what might work. Please check your Tru64 system documentation or contact your HP service representative for the precise instructions.)

The procedure described here is best performed by a system administrator or similarly trained person. The `vmunix` file created and copied at the end is the UNIX kernel. Modifying it incorrectly or corrupting it can have serious consequences.

1. Log in as root. You must be root (or have superuser privileges) to perform most of the following operations.

2. Change into the `/sys/conf/` directory.

```
prompt# cd /sys/conf
```

3. Edit the file whose name is the machine name, which we will call *MACHINE_NAME*, as follows. (Such a file typically exists after normal Tru64 Unix install. If the file does not exist, the System Administrator will have to create one.)

Change the line

```

dfssiz          2097152
to
dflssiz        8388608
or
dflssiz        16777216

```

The value of **dflssiz** may already be different than 2097152, which is 2 megabytes -- (* 2 (expt 2 20)). The new suggested values are 8388608, which is 8 megabytes and a good value for 32-bit Lisps, or 16777216, 16 megabytes and a good value for 64-bit Lisps.

4. Run /usr/sbin/doconfig as follows (recall that *MACHINE_NAME* is the file in */sys/conf* whose name is the machine name):

```
prompt# /usr/sbin/doconfig -c MACHINE_NAME
```

You should see output similar to:

```
*** KERNEL CONFIGURATION AND BUILD PROCEDURE ***
```

```
Saving /sys/conf/MACHINE_NAME as /sys/conf/MACHINE_NAME.bck
```

```
Do you want to edit the configuration file? (y/n) [n]: n
```

```
*** PERFORMING KERNEL BUILD ***
```

```
Working....Thu Oct 4 09:58:16 PDT 2001
```

```
The new kernel is /sys/MACHINE_NAME/vmunix
```

5. Copy the new kernel to / with a command like the following

```
prompt# mv /sys/MACHINE_NAME/vmunix /
```

6. Reboot the system.

5.3 Notes of increasing the default maximum stack size on HP-UX 11 machines

Increasing the default maximum stack size on HP-UX 11 requires modifying kernel configuration parameters, then rebuilding the kernel. This process should be done by the systems administrator.

The file `/stand/system` is the kernel configuration file. The `maxssiz` and `maxssiz_64bit` tunable parameters need to be increased to at least 32MB each. If you don't have any references to `maxssiz` or `maxssiz_64bit` in your `/stand/system` file, then you can simply add these lines to the bottom of the file:

```
maxssiz          (32*1024*1024)
maxssiz_64bit    (32*1024*1024)
```

Otherwise, you'll need to modify the existing parameters so that they express a value $\geq 32\text{MB}$.

After modifying `/stand/system`, the kernel needs to be rebuilt using the following command:

```
/usr/sbin/config -u /stand/system
```

The system will have to be rebooted for the changes to take effect.

5.4 Mac OS X notes

Allegro CL 8.0 is only supported on Mac OS X version 10.4 or later. It is not supported on versions 10.3 or earlier.

The Runtime Analyzer (see *runtime-analyzer.htm*) does not work properly on the 64-bit Mac OS X port (there is no problem on the 32-bit port).

Building shared libraries on Mac OS X

Building shared libraries on Mac OS X in *foreign-functions.htm* describes how to create a shared library suitable for loading into Allegro CL. We have determined that the `-flat_namespace` to the `ld` used

to create the shared library is necessary, as shown in the linked section.

5.5 Heap start locations

When building large new images, it is often useful to specify Lisp Heap and C-heap start locations. See the discussion of the *lisp-heap-start* and *c-heap-start* keyword arguments in *Arguments to build-lisp-image 2: defaults not inherited from the running image in building-images.htm*. Here are the initial locations (called `bases') in Allegro CL 7.0 as delivered. Values are Hexadecimal integers.

Operating System	Lisp base	C base
Tru64 32-bit	0x30000000	0x54000000
Tru64 64-bit	0x1000000000	0x2000000000
FreeBSD	0x40000000	0xa0000000
HP-UX 32-bit	0x10000000	0x64000000
HP-UX 64-bit	0x8000001000000000	0x8000002000000000
Linux (x86)	0x71000000	0xa0000000
Linux (PPC)	0x40000000	0x74000000
Mac OS X 32-bit	0x10000000	0x64000000
Mac OS X 64-bit	0x1000000000	0x2000000000
Windows	0x20000000	0x54000000
AIX 32-bit	0x30000000	0x64000000
AIX 64-bit	0x7000010000000000	0x7000020000000000
Solaris 32-bit	0x40000000	0x54000000
Solaris 64-bit	0x1000000000	0x1000000000
Solaris (AMD64) 64-bit	0x1000000000	0x1000000000
Linux (AMD64) 64-bit	0x1000000000	0x2000000000

6.0 Release Notes for the base Lisp

This main section contains three subsections (which have additional subsections): one on new features

([Section 6.1 New features in the base Lisp](#)), one on changes which are not backwards compatible and so may require code changes, ([Section 6.2 Non-backward-compatible changes in the base Lisp](#)), and one on miscellaneous changes ([Section 6.3 Other changes to and notes about the base Lisp](#)).

6.1 New features in the base Lisp

We have added a number of new capabilities to Allegro CL. Here we give links to the documentation of the new features.

The features described in [Section 6.1.1 Features added to Allegro CL 7.0 after the initial release of Allegro CL 7.0](#) were released originally as patches to Allegro CL 7.0, but after the initial release of Allegro CL 7.0. The features described in [Section 6.1.2 Major new features in Allegro CL 8.0](#) are new in the Allegro CL 8.0 release.

6.1.1 Features added to Allegro CL 7.0 after the initial release of Allegro CL 7.0

Various additions to Allegro CL 7.0 were added after the original release of Allegro CL 7.0. We list them here. All are part of release 8.0.

1. **New OLE/OCX interface:** a new, high-level interface to OLE/OCX on Windows was released in early September, 2005. The documentation is in the revised *ole.htm*.
2. **Update to an Allegro CL/SOAP API.** There was a significant new SOAP release which included the following enhancements:
 - CLOS wrappers for SOAP structures
 - built-in definitions for many Schema types
 - new warning hierarchy
 - new condition hierarchy
 - access to element attributes in SOAP messages
 - attribute values may be computed when message is composed
 - top-level multiRef handling
 - multiple SOAP Body elements
 - access to encoding and content-type headers
 - complete access to all the message components
 - user extensions to WSDL code generators

There are additional changes to SOAP in 8.0 not included in the 7.0 update.

3. **Blowfish encryption update:** The update (1) allows specifying the external format to be used when encrypting and decrypting; (2) adds support for CBC encrypting, for PKCS#5 padding, and zero (or null) padding for plaintext with no nulls; and (3) also adds new auxiliary functions **hex-string-to-usb8-array** and **usb8-array-to-hex-string**. See *Support for Blowfish encryption in miscellaneous.htm*.

In 8.0, the default value for the *external-format* argument of **blowfish-init** is changed to `:utf-8` (it was `:default` in release 7.0).

4. **New prolog release:** with new documentation *prolog.html*.
5. **New functionality for retrieving DNS TXT records.** The function **socket:dns-query** now accepts `:txt` as a value for the *type* keyword argument. When `:txt` is specified, the answer provided by **socket:dns-query** will be a list of one or more strings (because TXT records can be composed of multiple strings), or `nil` if there are no TXT records. See also *dns.htm*.
6. **ftp-stream-file-mod-time and ftp-file-mod-time now convert the remote-path argument to Unix syntax pathname naming; map-over-ftp-directory now works on Windows. ftp-file-mod-time and ftp-stream-file-mod-time both take a remote-path argument which is a pathname namestring. ftp functions only understand Unix pathname syntax, using forward slashes to delimit directories. Windows uses backward slashes to delimit directories and functions like cl:directory called on Windows returns pathnames with backward slashes in their namestrings. ftp-file-mod-time and ftp-stream-file-mod-time now convert those namestrings to Unix syntax using the new syntax keyword argument to cl:namestring (see cl:namestring in implementation.htm). Functions such as map-over-ftp-directory, which call ftp-stream-file-mod-time, now also work correctly on Windows.**
7. **New :syntax keyword argument to cl:namestring converts backslashes into forward slashes.** The **cl:namestring** function has a new keyword argument, *syntax*. The argument is ignored on Unix and Unix-like platforms. On Windows, when `nil` (the default), **cl:namestring** behaves as always and when `:unix`, **cl:namestring** converts back slashes in the namestring to forward slashes.

This feature is useful for ftp functions since ftp only understands Unix pathname syntax (using forward slashes). Functions like **map-over-ftp-directory** called on Windows generates pathnames of the files in an ftp directory, but these generated pathnames use Windows syntax (with backward slashes delimiting directories). In order for these pathnames to be used in calls to other ftp functions, such as **ftp-stream-file-mod-time**, they must be first converted to Unix syntax. Users writing their own mapping functions for ftp directories may find this added feature of **cl:namestring** useful. See *cl:namestring in implementation.htm*.

8. **Aodbc update:** this update provided the ability to set the non-blocking flag in a database. See **connect** and **db-non-blocking**.

9. **Oracle-direct update:** this update provided some additional functionality for the Oracle Direct interface. See *oracle-interface.htm*.
-

6.1.2 Major new features in Allegro CL 8.0

Other new capabilities to Allegro CL are listed here, with links to the documentation of the new features. Some of these features have also been released as patches to Allegro CL 7.0.

1. **AllegroCache:** AllegroCache is a high performance, dynamic object caching database system. It allows programs to work directly with objects as if they were in memory while in fact the object data is being stored persistently. AllegroCache supports a full transaction model with long and short transactions, and meets the classic ACID requirements for a reliable and robust database. See the AllegroCache documentation (PDF files), located in the **[Allegro Directory]/acache/doc/** directory: *acache.pdf* and *acachetutorial.pdf*.
 2. **:explain :inlining declaration:** there are various `:explain` declarations which tell the compiler to print out information about what it is doing or trying to do as it compiles. These declarations are described in *Help with declarations: the :explain declaration in compiling.htm* and also see the new *compiler-explanations.htm* document. The new `:explain :inlining` declaration provides information about why inlining of functions succeeds or fails. See *Inlining explanations in compiler-explanations.htm*. There is an example in that section.
-

6.1.3 Other new features in Allegro CL 8.0

1. **New -ee command-line argument and new function make-escaped-string:** this argument is the same as the existing `-e` command-line argument but it will additionally process its companion argument to convert character triplets of the form `%hh`, where *hh* is a two digit hex value, to the character whose char-code is that hex value. This allows you easily to specify as a companion argument values containing spaces, backslashes, etc.. Because the conversion is not done until Lisp is ready to process the argument, earlier shell processing will not have incorrectly resolved the special characters before Lisp sees them. See *Command line arguments* and *Further description of the -e and -ee command-line arguments*, both in *startup.htm*.

The new function **make-escaped-string** will create converted strings suitable as companion arguments to `-ee`.

2. **New macro with-native-strings*.** This new macro allows nesting of calls to **with-native-string**.

3. **New function `system:constant-value`:** the function `system:constant-value` takes a form as an argument, and if the form has constant value, returns its value. An environment can be passed as an optional argument.
 4. **New function `sys:ensure-portable-walking-environment`.** `sys:ensure-portable-walking-environment` returns an environment suitable for portable code walkers to use entirely within the ANSI Specification of Common Lisp.
 5. **New second optional argument to `cl:macroexpand` and `cl:macroexpand-1`.** This second optional argument allows more choices in macroexpansion in different environments. See *cl:macroexpand* and *cl:macroexpand-1* in *implementation.htm* for details.
 6. **New macro `mp:with-message-interrupts-disabled`:** on Windows, even though a process may be waiting, it still typically watches for messages and processes operating system messages that require a response (because Windows expects threads to do this). However, there are some rare circumstances where this message handling should be suppressed. The macro `mp:with-message-interrupts-disabled` will do this.
-

6.1.4 Multiprocessing and OS threads in Allegro CL 8.0

The multiprocessing model was significantly changed in release 7.0. There are not major changes in release 8.0 compared to 7.0, other than bug fixes and internal improvements. See *The new model for multiprocessing and OS threads in Allegro CL in version-70-release-notes.htm* details of the 7.0 changes.

process-suspend-hook and process-resume-hook will go away soon

Currently a programmer can set a process's **process-suspend-hook** and be sure that whenever the process's execution is interrupted and another process allowed to run, the hook will get executed. Similarly, when a process regains control after another process has been running, its **process-resume-hook**, if any, will be run before its execution continues.

Unfortunately, suspend-hooks and resume-hooks cause problems for native thread implementations and other proposed enhancements in the multiprocessing model.

Therefore, these features will likely be removed in the next release. If you use these features, please contact support@franz.com so we can assist you in providing alternatives when these features are removed.

6.2 Non-backward-compatible changes in the base Lisp

1. **ACL_STARTUP_DEBUG support removed.** Until support was removed with a 7.0 patch, Allegro CL would examine the environment variable `ACL_STARTUP_DEBUG` on startup and print startup information if it was set. This is no longer supported.
2. **Second (non-standard) argument to `cl:sleep` removed.** This optional argument to `cl:sleep`, which was documented as being for internal use only, has been removed. Any code which called `cl:sleep` with two arguments should be modified to call it with one argument only. Also, only non-negative arguments are accepted by `cl:sleep`. Passing a negative argument causes an error.

`cl:sleep` is now effectively `mp:process-sleep` (with the `whostate` argument hardwired to "sleeping"). In earlier releases, there was a function `excl:lisp-sleep`. That function was removed in release 7.0. See *cl:sleep and minimum sleeping time* in *multiprocessing.htm*.

3. **Change to the default value of the external-format argument to `blowfish-init`.** The default value for the *external-format* argument of `blowfish-init` is changed to `:utf-8` (it was `:default` in release 7.0). It is unlikely that this change will affect any code. The argument was only added by a patch in September, 2005. The sensible default is `:utf-8` (which is the default for `blowfish-encrypt` and `blowfish-decrypt`) but the behavior prior to the argument being added was to use `:default` so the patch kept the same behavior. In 8.0, the more sensible default is used.
4. **Environment name switching: the names of environments have been changed.** What was in 7.0 a `:compilation` environment is now a `:compiler` environment and what in 7.0 was a `:compilation-walking` environment is now a `:compilation` environment. See *environments.htm*.
5. **Environment variable name change:** the variable `sys:*compile-file-environment*` is deprecated, and the preferred name is now `sys:*compilation-unit-environment*`. The old name does not have a value but does name a symbol-macro which causes the new name to be used.
6. **Environment function replaced:** the function `sys:make-compile-file-environment` is deprecated. Use the new `sys:make-compilation-unit-environment` instead. Calls to `sys:make-compile-file-environment` (which took no arguments) can be replaced with calls to `sys:make-compilation-unit-environment` with no arguments. `sys:make-compilation-unit-environment` accepts an optional argument, which, when true, returns an environment suitable for `compile` and when nil (the default, returns an environment suitable for `compile-file` (as `sys:make-compile-file-environment` did).
7. **The default value of `*open-rename-function*` now writes file using `*default-pathname-defaults*`.** In earlier releases, when a file was written because `open` was called for writing to an existing file with `:if-exists` specified to be `:rename`, the function which is the initial value of `*open-rename-function*` would use the current directory to fill in missing parts of the renamed file's directory location, rather than using `*default-pathname-defaults*`. Since `open` used `*default-pathname-defaults*`, the result could be the new file and the backup in different directories when the value of `*default-pathname-defaults*` was different from the current directory (as returned by `current-directory`). In release 8.0, the renaming function does use `*default-pathname-defaults*`. This is a non-backward-

compatible change although we suspect anyone who has noticed the behavior has written a renaming function to behave as the new default renaming function now does. See `*open-rename-function*`.

8. **Revised handling of nested conditionals, new variables.** The handling of nested `#+/#-` conditionals has been revised. A nested conditional is a form like `(list #+allegro :allegro #-allegro #+foo :foo #-foo :default)`. Until this change, Allegro CL would evaluate that form (assuming `:allegro` is a feature and `:foo` is not) to `(:allegro :default)`. With the change, it now follows other implementations and returns `(:allegro)`. Also, before the change the form `(list #+allegro :allegro #-allegro #+foo :foo)` returned `(:allegro)` in Allegro CL but errored in other implementations. It now errors in Allegro CL as well.

The issues are complicated. There are two new variables that affect this behavior: `*sharp-plus-de-facto-standard-compatible*` and `*warn-on-nested-reader-conditionals*`. The section *The issue of nested conditionals in Allegro CL in implementation.htm* discusses nested conditionals in detail. See that section and the descriptions of the two variables for more information.

6.3 Other changes to and notes about the base Lisp

1. **New additional optional argument to `make-random-state` allows seeding with outside random values.** `make-random-state` has an additional optional argument `seed` that allows specifying a value to use to seed the random number generator (it is effective only when the `state` optional argument is specified as `t`). See *cl:random and cl:make-random-state in implementation.htm*.
2. **`build-lisp-image` now has a `:require-search-list` argument.** This argument allows specifying a value for `*require-search-list*` (which instructs an image where to look for modules). The value defaults to the value of `*require-search-list*` in the running image. See *building-images.htm*, where `build-lisp-image` is fully described.
3. **Certain `build-lisp-image` arguments can now get default values from environment variables.** The `c-heap-size`, `c-heap-start`, `lisp-heap-size`, `lisp-heap-start`, `newspace`, and `oldspace` keyword arguments to `build-lisp-image` can get their default values from environment variables (named `ACL_BUILD_[arg-name-with-underscores]` -- e.g. `ACL_BUILD_C_HEAP_START`). See *Arguments to build-lisp-image 3: defaults taken from environment variables in building-images.htm*.
4. **Changes to the sax parser.** There have been some minor changes to the sax parser, described in *sax.htm*. The class `lxml-parser` has new `normalize` and `package` slots and `parse-to-lxml` has new `normalize` and `class` keyword arguments. The `package` argument in `parse-to-lxml` now defaults to the `package` value in the class that is the value of the `class` argument. Normalizing allows an element that contains only string content to appear as one contiguous string (at the cost

of additional consing). Finally, the **pxml-version** now accepts `:query` as a value for its optional parser-type argument: when specified, type of the enabled version of the **parse-xml** operator is returned.

5. **:nat and :unsigned-nat foreign types.** This change was actually added in release 7.0 but not previously documented. These types are described in *The Syntax for Foreign Types* in *ftype.htm*. "Nat" stands for "natural" and is intended to be the size of integer which fits into the natural word size of the machine - 32 bits on a 32-bit lisp and 64 bits on a 64-bit Lisp.
6. **New compiler switch trust-table-case-argument-switch.** `trust-table-case-argument-switch` is true when speed is 3 and safety is 0. If a case statement is suitable for compilation to a table-driven case statement (see the description for details), which is faster but unsafe, it will be compiled that way when this switch is true.
7. **New function gets the environment of Lisp.** `environment` return a list of the original environment when Lisp was started.

6.4 Base Lisp platform-specific information

There are no entries at this time. Information may be placed here in documentation updates after the initial Allegro CL 8.0 release.

7.0 Release Notes for CLIM

The CLIM manual has not been updated (other than minor corrections) for the 8.0 release. There have been no significant changes to CLIM functionality compared to 7.0, though there have been bug fixes and performance enhancements.

(Repeated mostly from 7.0 Release Notes.) The documentation for CLIM is in an online PDF file, *clim-ug.pdf*.

On Linux and FreeBSD, Allegro CLIM works with Open Motif 2.1 and 2.2 (2.2 was released in early 2002). Open Motif 2.2 is available at no charge from www.openmotif.org. Allegro CL 8.0/Allegro CLIM will work with either version. Redhat Linux 7.2 is supplied with Lesstif, a version of Motif that **does not work with CLIM**. See [this faq item](#) for information on how to install Redhat Linux 7.2 without lesstif and how to uninstall lesstif if already present.

On Mac OS X 10.3, OpenMotif Motif 2.2.2 is required. CLIM will not work with version 2.2.0.

Certain CLIM demos on Solaris 64 bit give segv or otherwise fail when displaying over the network, language environment must be set to C. This is a problem when running any Sun GUI (such as the CDE environment or the Gnome 2.0 interface) on a Solaris64 machine. When bringing up the environment, set the language/locale to "C". (On the login screen, there's an "Options" button, which displays a menu that has a "Languages" submenu. Choose "C". Note: the default value is typically "en_US".) The "C" setting can process all of the 64-bit font sets. However, difficulties arise when displaying over the network. If you are displaying on the monitor of the machine running Allegro CL (and CLIM), the demos work as long as the language/locale is "C". They typically do not work when displaying on a monitor over the network. As of this writing, there is no fix for the problem.

8.0 Release Notes for Common Graphics and the IDE

There are few significant changes to Common Graphics and the Integrated Development Environment in 8.0 (those few are described in [Section 8.1 Significant changes in Common Graphics](#)). Work on Common Graphics and IDE support on Linux has progressed. Allegro CL 8.0 on Linux includes a preliminary version of Common Graphics and the IDE (similar to what was released for Allegro CL 7.0). Note that some destabilization may have occurred due to reworking some pieces of code in order to share more code with the new GTK port of Common Graphics and the IDE for Linux/Unix.

There were many significant changes in release 7.0. Users unfamiliar with those changes should look at the *version-70-release-notes.htm*.

The [first subsection](#) describes changes to Common Graphics and the IDE that are non backward-compatible. Please review this section and make whatever necessary changes to your code to obtain the desired behavior in release 8.0.

The [second subsection](#) describes other changes to Common Graphics and the IDE. These should not require code changes (please tell us if any do, because that may indicate a bug), but note that certain function and argument names have been deprecated in favor of new names, and that new code should reflect these changes, and old code should be revised at some point.

The section [Section 8.5 IDE release notes](#) and its subsections provide information about the IDE.

8.1 Significant changes in Common Graphics

New OCX control class and associated functionality

The new macro **def-cg-ocx-control** defines a Common Graphics dialog-item class for one of the OLE controls that was defined by a **def-ole-linkage** form. A newly-defined control could be added to the IDE's Components Toolbar as usual by calling **add-to-component-toolbar**. A couple of known remaining problems: (1) The value of certain properties will be an interface object, and so it is not feasible to modify the value at this time. (2) There is not yet a way to return a value in an "in-out" parameter of an OLE method.

Two new controls: lamp and html

- **New lamp widget class.** The `lamp` widget class implements the lamp widget. There are a number of associated generic functions implementing aspects of lamp functionality.
- **New HTML control.** There is a new `html-widget` dialog-item plus an `html-browser` dialog for adding HTML browsing capabilities to Common Graphics apps. It uses Microsoft's WebBrowser OLE control, which is also used by Internet Explorer. Unfortunately, there appears to be no way to search for text in the displayed HTML.

8.2 Non-backward-compatible changes in Common Graphics

1. **All cg/ide packages are locked.** This makes things more like release 6.2 and earlier when the symbols were all defined in the `cg` package, which was locked. In 7.0 there was less locking because the child packages had been created but inadvertently not locked. This may cause user methods that specialize on built-in classes to cause package lock warnings or errors that did not occur in 7.0. Such an error generally indicates that you should be specializing on your own subclass rather than on a built-in class.
 2. **draw-curve renamed draw-bezier-curve.** The new name is more descriptive and `draw-curve` too easily conflicted with user function names. See **draw-bezier-curve**.
 3. **range-equality-test deprecated:** the function which is the value of **range-equality-test** is no longer used by Common Graphics code. (The function was used to compare old and new values of the **range** of a dialog-item to see whether there was really a change.) **on-change-test** is now used for that purpose.
 4. **save-selection-when-unfocused deprecated:** the Windows code that implemented `save-selection-when-unfocused` has not been supported by Microsoft for some time. There is no practical way to implement the behavior when `nil` in GTK. The default is true (the text stays selected). Specifying it to be `nil` may or may not be effective and is not recommended.
 5. **The "context menu key" (`vk-applications`) will now call pop-up-shortcut-menu by default.** Previous the default behavior was to do nothing. See `vk-applications`.
-

8.3 Functionality to handle differences between Windows and GTK

Common Graphics and the IDE now run on Windows and Linux with GTK. Certain differences between the two operating systems and windowing systems mean that some things do not work the same in Windows as in GTK and vice versa. The functionality listed here allows tries to handle the differences. Some of the variables/operators/etc. only available on one of the two platforms.

Mozilla support

The system needs to know the directory where the GTK control supplied by Mozilla to support the `html-widget` resides. (On windows, the widget is implemented in another way.) It may not be practical to search for the location automatically. See **mozilla-library-path**, `*mozilla-library-path*`, and **find-mozilla-gtk-path**.

Event handler processes

Because Linux currently does not use native threads and Windows does, on Windows, all windows handle events in their own thread associated with the window while on Linux/GTK, there is one event-handling thread. **default-application-window-subkey**, `*use-single-cg-event-handling-process*`, and `*single-cg-event-handling-process*` allow distinguishing behavior when necessary. **cg-process-wait** should be used on GTK instead of **process-wait** in event handlers. See also **process-pending-events-if-event-handler**.

Miscellaneous

- **Linux has a built-in hand cursor**, which is the value of `hand-cursor`. That variable is `nil` on windows. (`find-cursor :hand-cursor`) works correctly on both,
- **Linux pixel size issues**: because Linux/GTK seems not to have a reliable notion of pixel size, the functions **pixels-to-points**, **points-to-pixels**, and **font-pixel-height** are provided. They work on Windows but are usually not necessary.
- **Timer resolution on Linux/GTK**: see `*cg-timer-resolution*`.
- **Check marks in menus: on Linux/GTK**: you must specify at creation time whether a menu-item will ever be checked. See **checkable**.
- **Modal dialogs**: when displaying a modal dialog, the owner window is usually disabled. On GTK, this make take too long. `*modal-dialogs-disable-owner*` handles this.
- **f10 key**: it requies special handling on Linux.GTK, see **handle-f10**.
- **Scroll increments**: the new function **scroll-increment** is needed on Linux/GTK. It works on Windows but there is essentially the same as **user-scroll** (which does not work on Linux.GTK).
- **Using the IDE parent-window**: issues of mixing windows from various applications are more complex on Linux/GTK than on Windows. See the functions **use-ide-parent-window**, **maximize-**

ide, **ide-exterior**, and **top-ide-window**, and the classes `ide-parent-window`, `ide-child-window`, and `ide-owner-window`. The IDE menu item **View | Maximize IDE** (Linux/GTK only) is related.

8.4 Other changes in Common Graphics

1. **New variable `*show-console-on-standalone-error*`**. The variable `*show-console-on-standalone-error*` (actually added in a 7.0 update) controls whether the console window is automatically shown when an error is signaled in a standalone Common Graphics application that was generated from a project in the IDE.
2. **New function `special-windows-directory`**. The function `special-windows-directory` determines the paths of certain Windows directories (such as the **My Documents** directory).
3. **New keyword argument to `load-pixmap`**. The function `load-pixmap` has a new *make-unique-name* keyword argument to ensure that the function does not create a name for the pixmap (from its file name) that is the same as the name of a cached pixmap.
4. **`on-print` event handler can now be nil**. The `on-print` property of a widget may now be set to `nil` like other event-handlers, in which case Common Graphics will default the value to **princ-to-string**. This avoids confusing errors that could otherwise happen if a user assumes that this property could be nil like other event-handlers and then CG funcalls `nil` on an object to print.
5. **transparent-pixel-value argument to `generate-mask`**. The *transparent-pixel-value* argument to `generate-mask` may now be `nil`. When it is, it takes the value of the upper-left pixel of the pixmap as the value to make transparent.
6. **Multi-picture-buttons and masked pixmaps**. `multi-picture-buttons` now handle masked pixmaps. Previously if a pixmap that has a mask was specified as the pixmap of a `button-info`, the transparent area of the mask was not drawn in the `multi-picture-button`'s background color.
7. **Autosizing list-views**. The `list-view` function **auto-size** will no longer truncate a column header string if all of the strings in the body of the column are shorter than the header string.
8. **New `tabs-are-draggable` property**. If you turn on the new **tabs-are-draggable** property of a `tab-control`, and the **single-line** property is also true, then the user can rearrange the tabs within the `tab-control` by dragging after clicking left on a tab. In the IDE, the **Editor** uses this to allow you to change the order of the editor buffers. The **Debug Window** also allows you to change the order of any additional listeners that you've created there with the **View | New Listener** command.
9. **New variable `*event-loop-processes*`**. `*event-loop-processes*` is a list of all processes that are currently inside a call to `event-loop`, and which therefore can handle Windows messages.
10. **New function `message-box`**: the `message-box` function displays a simple message dialog using the platform's low-level utility to do so.
11. **Change to `font-equal`**: the function `font-equal` will now return true if the family field of one font is `nil` but the other is not. This compensates for the fact that the family field (which typically

- has no effect) sometimes gets "filled in" and sometimes it does not.
12. **move-window-into-parent applied to top-level windows:** the function **move-window-into-parent** now works for top-level windows.
 13. **The initial-name argument of ask-user-for-new/existing-pathname. ask-user-for-existing-pathname and ask-user-for-new-pathname** now accept pathname objects as well as namestrings as values for the *initial-name* argument.
 14. **Default for link-color:** the **link-color** property defaulted to blue for a `rich-edit` dialog-item but to red for a `rich-edit-pane`; now it is blue for both.
 15. **delete-command undeprecated:** **delete-command** is now undeprecated because it passes the command down to a descendent window (like **copy-command** and the others) before it calls **delete-selection**. (It was deprecated in 7.0.)
 16. **Clicking on a text-edit-pane whose parent is a child window.** Due to an apparent Windows quirk, clicking on a `text-edit-pane` whose parent is a child window did not expose the parent window. A Common Graphics method has been added to cover for that anomaly.
 17. **draw-string-in-box obeys vertical justification when wrap-p is true:** **draw-string-in-box** now handles the case where *wrap-p* is true and vertical-justification is `:center` or `:bottom`. The text had always been top-justified when *wrap-p* is true.
 18. **grid-widget cell-box obeys vertical justification:** `grid-widget` check-box cells (see `checkbox-column-mixin`) now obey its **cell-vertical-justification** method (previously it always acted as if the value was `:top`).
 19. **Bug with grid read-cell-value interacting with AllegroCache is fixed:** a bug was fixed where drawing a grid cell could break badly if its **read-cell-value** method accessed an Allegrocache slot in client-server mode. Waiting on the socket stream for a reply had allowed further messages to be handled, which could make a nested access to the object during scrolling, for example. **read-cell-value** and **write-cell-value** now suppress handling further messages until they return. An application's custom **redisplay-window** method could have a similar problem, in which case it should wrap a call to **mp:with-message-interrupts-disabled** around the code that does a process-wait.
 20. **sort-grid is now robust against missing widget-window:** **sort-grid** no longer breaks if the `grid-widget` has no `widget-window` at the time. The **sort-column** property may now be specified with a `:sort-column` initarg. A `grid-column` name may now be passed to **sort-grid** as documented, rather than only a `grid-column` object.
 21. **New generic function cell-widget:** a `grid-widget` application may define methods on the new generic function **cell-widget** to use the pre-defined cell widgets in arbitrary cells rather than only in particular grid columns for each type of widget.
 22. **New button-fills-cell property for static-text-and-button-column-mixin.** The `static-text-and-button-column-mixin` class has a new **button-fills-cell** property that causes the button to fill the grid cell, with the text inside it.
 23. **Function that is the value of the button-function property of static-text-and-button-column-mixin instances now takes six arguments.** If an application uses the `static-text-and-button-column-mixin` (for `grid-widget` columns), then the function that is the value of the **button-function** property should now take two additional arguments for the row-number and column-number (making six arguments in all). For backward compatibility, the Common

Graphics code will still handle old functions that do not take these arguments, but all **button-function** functions now should officially accept them. So a button function could now be defined like this:

```
(defun my-button-function
  (grid-widget data-object row column row-number column-number)
  ...)
```

Or this alternative would work in both 8.0 and 7.0 and earlier:

```
(defun my-button-function
  (grid-widget data-object row column
    &optional row-number column-number)
```

24. **New function make-message-window and macro with-message-window.** **make-message-window** creates a window suitable for displaying as a message window. **with-message-window** creates and displays such a window while code is run.
25. **New function import-pixmaps.** The new function **import-pixmaps** makes it easy to convert a set of .bmp pixmap files into lisp source code for embedding the pixmaps into the single image file of a delivered application.
26. **A Common Graphics window may now be larger than the screen if desired.** After creation, an application's **track-limits** method (if any) can constrain the size of the window as before.
27. **Rich-edit functionality:** the functions **rich-edit-get-color**, **rich-edit-get-font**, **toggle-bold**, **toggle-italic**, **toggle-underline**, **toggle-bullets**, **left-justify**, **center-justify**, **right-justify** are used by **rich-edit-multipic** widgets and menu-bars on **rich-edit-dialogs**. (These were available but not documented in 7.0.) Also see **funcall-menu-item-with-rich-edit**.
28. **draw-to-x-y:** like **draw-to**, **draw-to-x-y** draws from the current position to a location, but the location is specified with coordinates rather than a position.
29. **drag-and-drop-mouse-moved:** allows added code to be run when the mouse is moved during a drag.
- 30.
- 31.

8.5 IDE release notes

1. **The IDE Help facility now can use an instance of the new CG html-browser dialog**, to display IDE help inside the IDE rather than in a standalone HTML browser program using **invoke-private-html-dialog**. You can return to the old behavior if needed by setting the new **use-cg-html-browser** ide configuration option to **nil**. In the new browser you can, for example, select a code example and then invoke the IDE's **Tools | Incremental Evaluation** command on it

directly, rather than first copying the example code to the editor. One disadvantage is that there is no string search in the CG HTML control (because Microsoft's WebBrowser OLE control does not appear to provide it).

2. **New use-cg-html-browser configuration option.** When the **use-cg-html-browser** option is true, help is displayed in the Common Graphics `html-browser`. This is more reliable and allows using IDE menu commands directly on the help page but (because of apparent limitations of the associated OCX widget) does not provide a string search capability. If you try to search, you will get a dialog that allows you to display the text in a third-party browser which supports such searches. If you set the value to `nil`, then the third party browser will be used all the time.
3. **Changes to the behavior of save-options-to-user-specific-file on Windows:** the function **save-options-to-user-specific-file**, which allows personalized *prefs.cl* files (with personal options settings) on multi-user machines, now uses Windows-specific directories like My Documents or Personal on Windows, rather than using the value of a HOMEDRIVE or HOMEPATH environment variable. See also **special-windows-directory** which determines the path of such directories. Also the default value of **save-options-to-user-specific-file** is now true.
4. **Trace dialog faster, can restrict function traced:** the **Trace Dialog** displays several times more quickly than in earlier releases. The **inhibit-trace-for-object** option allows trace to be disabled in the event-handler process of specified windows.
5. **IDE configuration option ide-html-history: ide-html-history** allows saving of previously-viewed HTML pages.
6. **Inspector displays changed property values in a bold font.** A property value will now be displayed in a bold font in the inspector if it is not equivalent to the programmatic default value for that property. This highlights values that you've changed, as well as some sample values that you probably want to change for widgets placed onto forms. A read-only value will never be bold. A handful of properties are never shown bold because they generally don't have meaningful default values and are more distracting than useful when made bold; these exceptions include the properties **child-p**, **device**, **tab-position**, **left**, **top**, **width**, **height**, **dialog-items**, and **pixmap-use-handle**.
7. **The File | Save command is now implemented for the Runtime Analyzer Results dialog.** You can now save displayed analysis outline to a text file as well as saving the history. See the description of the **Runtime Analyzer Results dialog**. There are now two buttons on that dialog: the new **Save to File** and the older **Save** button, now renamed **Save to History**.
8. **Save command on Class Browser.** The **Save** command is implemented for the **Class Browser** outline, to save a class hierarchy as indented text to a text file.
9. **The Class Browser dialog now uses a multi-picture-button rather than a tab-control.** This was mostly for compatibility with the GTK port, where the tab-control cannot have multiple rows of tabs. Though the set of choices is the same, the new buttons display icons rather than text, so you may need to learn which button is which by holding the mouse cursor over the buttons until their tooltips appear.
10. **More things can be searched backwards.** Searching backward now works in item-lists, list-views, the inspector, and the tree grapher. (This had worked only in text windows like the IDE editor.)
11. **Help on widget subclasses:** invoking help on a user-subclassed widget on a form will find the

- help for the built-in superclass.
12. **Help command improvements:** the Help command is now included reliably on the right-button shortcut menu of the inspector.
 13. **Shortcut menus.** The shortcut menu for a form window is now accessible on the title-bar and menu-bar, in case widgets cover the entire form interior.
 14. **Changing the default package (in the Project Manager dialog)** of an existing project should now be less troublesome, but it is still far better to decide on the package name when first setting up the project (especially before creating any form windows).
 15. **Handling non-existent package in a project:** if you type the name of a non-existing package into the Project Manager dialog, it automatically creates a package of that name. This new package had used the `excl` and `cl` packages, even though the code that is generated for the project does not make the project package use those packages. That could lead to errors loading the project into a later lisp session when the project package would no longer use those packages as it had initially. So the project manager no longer makes the automatically created package use those packages.
 16. **Source code colorization.** Source code colorization is no longer done in an IDE editor buffer if the buffer has been saved to a file and the file does not have one of the common extensions for lisp source code (`.cl`, `.lsp`, or `.lisp`).
 17. **open-files-in-gnu-emacs mode:** when using the **open-files-in-gnu-emacs** mode, the IDE's Save All command will tell emacs to save all of its unsaved buffers.
 18. **Compiling a project:** when you compile the current project, if it has multiple subprojects and the multiple subprojects all in turn have a common subproject, then that common subproject will be compiled only once rather than once for each subproject that has it as a subproject.
 19. **Some symbols no longer exported from cg package.** A handful of overly-short IDE symbols were exported from CG to avoid a problem with an app using the symbols as local variables that end up in the IDE package, which is not available in a standalone application. The symbols are still exported from the `ide` package as before, and most of them are project accessor functions that are probably rarely used by users. The symbols are: `project`, `projects`, `libraries`, `module`, `modules`, `module-p`, `verbose`, `loaded`, `partners`, and `followers`.
 20. **Optional arguments to full-compile-project:** the optional arguments of **full-compile-project** are now keyword arguments to match the similar function **compile-project**. Usually you would use the menu command rather than calling this function programmatically, but if you do call the function with any of its optional arguments then you will need to change the call to pass keyword arguments instead.
 21. **Lost IDE windows:** if an IDE window gets "lost" by being moved outside of the visible screen area (perhaps programmatically), then the Window List dialog's Show button will move it into view (in addition to selecting the window as usual).
 22. **Moving to the beginning of a definition in the IDE editor:** in the IDE editor, the gesture to move to the beginning of the definition (`alt-uparrow` in host mode or `control-alt-A` in emacs mode) will now move to the beginning of the previous definition when the text cursor is already at the beginning of a definition. If there is no definition that begins before the text cursor, it will move to the top of the file. Previously it did nothing in these cases. The new behavior emulates the similar command in emacs.

23. **Missing project files:** when there are missing project files while opening or running a project, you will now be given the opportunity to locate the files with the file dialog.
24. **Missing modules in a project:** when generating a standalone application from a project, and it fails because a needed CG module was not included, the error dialog will now mention the particular module and offer to automatically add it and try the build again.
25. **Names of editor buffers:** it's easier to find the name of an IDE editor buffer that you're looking for when using the editor's pop-up menu of all buffers (control-B in the default editor mode). It now displays the pathname-name by itself at the left, and sorts by that name rather than by the whole pathname.
26. **New IDE configuration option `use-ide-parent-window`.** There's a new configuration option to use a single parent window for the IDE, called **`use-ide-parent-window`**. It's on by default on GTK because owner window support is not good there, but is NOT on by default on Windows because child windows cannot have menu-bars in Windows, and form windows therefore will not show their menu-bars. The new option has not been tested rigorously on Windows, but you can change it by evaluating `(setf (use-ide-parent-window (configuration *ide-system*)) t)` in the IDE, and then restarting the IDE.
27. **Project package now used when a project is run.** When doing a **Run | Run Project** or running a standalone application that was made from a project, the value of `*package*` will initially be the project's default package. (This is not done, however, by **Run | Run Form** because the form runs asynchronously in an existing IDE Listener process.) Formerly the package property of a project was used only for writing an **in-package** form to any new code files that the project system generates.
28. **Help menu now has Write a Bug Report item.** The **Help** menu now has a **Help | Write a bug Report** item for use when there is no backtrace showing (the **Debug Window after an error** also has a bug report button). The new menu item lets you choose any subset of processes, to add their backtraces to the bug report.
29. **maker-function now uses interior argument instead of exterior argument.** When a form window in a project is saved, it will now generate the **maker-function** so that it has a new *interior* parameter with a default value that is equal to the form's current interior, and the *exterior* parameter will no longer have a default value. This will cause the running window to be created (by default) with the same interior on machines or operating systems where the title bar height or border thickness are different. This avoids, for example, buttons at the bottom of a dialog being partly clipped off when a generated application is run on a different machine that has a larger title bar height. One possible drawback to this design change would be if an application positions multiple windows that are made from forms so that their exteriors are adjacent or spaced by a certain amount. In that case, the application could pass the *exterior* argument to the **maker-function** explicitly. Or it could create each window in `:shrunk state`, then move or resize it as needed once its exterior is known, and then expose it.
30. **New navigator examples for cell-widget and windows with multiple panes.** See the Examples tab of the **Navigator Dialog**.
31. **Some changes to alignment lines and snapping on forms.** When dragging a widget (or block of widgets) on a form, alignment lines and snapping (see **snap-to-components**) will no longer be done between the top of a moving widget and the bottom of a non-moving widget, or vice versa,

or between a left side and a right. Alignment and snapping are now done only between two top sides, two bottom sides, two left sides, or two right sides. The cases that were removed are generally not useful and cluttered things up. Also, the alignment lines should now reliably be drawn so that they extend to encompass all of the widgets that are currently aligned with the moving widget(s), rather than only some of them. Alignment lines and snapping will also be done between a moving widget (or block of widgets) and their former position, to make it easy to move the widgets only horizontally or only vertically.

8.5.1 Opening projects from releases prior to 8.0

Allegro CL 7.0 projects should open without problem in Allegro CL 8.0. For projects from releases prior to 7.0, see *this section of [version-70-release-notes.htm](#)*

9.0 AllegroStore not available in Allegro CL 8.0

AllegroStore is not available in Allegro CL version 8.0. Users who wish to continue using AllegroStore should use version 7.0.

There is a new database product, AllegroCache. See [Section 6.1.2 Major new features in Allegro CL 8.0](#).

10.0 Release notes for SOAP

Major Additions or Revisions:

- **Added support for nillable attributes** for SOAP elements. The `nillable` option, when `non-nil`, adds the `"xsd:nillable='true'"` attribute to the element (see *Element Definition* in *soap.htm*).
- **Recognize all Schema components** (some Schema features are still ignored, but will no longer signal errors).
- **New generic function `wSDL-generate-code`** allows application programmers to modify the code generated by `make-client-interface` and `make-server-interface`.

Other changes include:

- The generic function **define-soap-element** now has a *nillable* keyword argument which marks elements as nillable.
 - **encode-wsdl-file** has a new *target-package* keyword argument.
 - New function **soap-new-environment** resets the SOAP environment.
 - New function **soap-make-fault** creates an encoded fault instance.
 - New *stream* argument to **decode-wsdl-source** allows input to come from a stream.
 - New *xml-syntax*, *class*, and *init* keyword arguments to **decode-wsdl-file**, **decode-wsdl-source**, **decode-wsdl-string**, and **decode-wsdl-at-uri**. *xml-syntax* allows specifying whether to insist on strict compliance with the standard. *class* allows specifying a subclass `wsdl-file-connector` for the connector. *init* allows specifying make-instance arguments when creating an instance of the subclass.
-
-

11.0 Release notes for Allegro RPC

There are new RPC functionality allowing users to manage multiple Lisp images. See *Running several communicating Allegro CL images* in *rpc.htm*.

The *re-connect* keyword argument to **make-rpc-client** can now have the value *:connect*. If specified, when a re-connect attempt fails, the function will try a new connect. **rpc-open-client** is also affected since it may use clients made by **make-rpc-client**.

12.0 Release notes for jLinker

Jlinker in ACL 8.0 adds significant new functionality. A native Java interface allows the Lisp and Java parts of an application to share the same address space. Calls between Lisp and Java are comparable in speed to foreign function calls.

Arguments in calls to Java are converted using the information in the Java method signature. Consequently, calls to **make-immediate-object** are now redundant (but harmless).

New Jlinker Native interface (JNI)

There is now JNI support in the Jlinker. The new interface is described in the *jlinker.htm* and can be used instead of the socket interface. The jlinker JNI interface depends on the availability of the **libjvm**

shared library in the Java installation. We have tested the jlinker JNI interface on some of the machine and operating system combinations where we have found a suitable Java distribution.

The status on various platforms is as follows:

- Windows: tested and working with Java 1.4.2 or later
- Linux x86 - tested and working with Java 1.4.2 or later (32-bit ACL and Java).
- Solaris - tested and working with Java 1.4.2 or later (32-bit ACL and Java)
- Mac OS X - Native (JNI) mode is disabled.
- Other UNIX - we have not located a suitable Java, hence not tested.

Non-backward-compatible change

1. **Change to initial value of `*jlinker-error-p*`:** the default value of the variable `*jlinker-error-p*` is now `t`. In previous versions, the default value was `nil`.

New functionality

The following variables and operators are defined:

- **jclass-paths** (function)
- **jget-properties** (function)
- **jget-property** (function)
- **with-java-environment** (macro)
- `*jlinker-init*` (variable)
- `*jni-library*` (variable)

jLinker documentation is in *jlinker.htm*.

Other jlinker changes

- **New keyword arguments to `jlinker-init`.** `jlinker-init` has new keyword arguments (*classpath* and *jar*) to specify the classpath and the `jlinker.jar` file location. Jlinker no longer modifies the CLASSPATH environment variable.
- **On Windows, Jlinker can examine the Windows Registry to locate Java components.** In typical installations, the user does not need to customize anything in order to use Jlinker.

13.0 Release notes for AllegroServe

No notes at this time

14.0 Release notes for IMAP and SMTP, XMLutils

No significant changes to these utilities

15.0 Release notes for The Emacs/Lisp interface

No significant changes.

16.0 Documentation modifications in Allegro CL 8.0

The format of the documentation in 8.0 is similar to that of 7.0. There is a new essay file *compiler-explanations.htm* which discusses the `:explain compiler` declaration, which has been enhanced in 8.0. (That material used to be in *compiling.htm*.)

17.0 Availability of CLX for Allegro CL

CLX (Common Lisp X) provides an interface between Common Lisp and the X window system. All versions of Allegro CL include a compiled version of CLX with the distribution. The *fasl* file is *code/clx.fasl*, loaded by evaluating `(require :clx)`. The Allegro CL products CLIM and Allegro Composer use CLX. Users wanting low-level access to an X server in Lisp may also want to use CLX. CLX is not supported by Franz Inc.

The sources to CLX are supplied with the regular Allegro CL distributions in the *contrib/clx/* directory. Note that during installation, you are asked whether you wish to install the *contrib/* directory and the default is not to install it. The *contrib/* directory is not included in the Trial distribution, but Trial users

can download the CLX sources from the Franz Inc. website as described next.

The sources to CLX are also available on the Franz Inc. web site (www.franz.com), at the location <ftp://ftp.franz.com/pub/clx/>.

18.0 Release notes for Orblink

No significant changes.

Appendix A: Tightening of ANSI Conformance in Allegro CL

There were many changes in release 7.0 tightening Allegro CL's compliance with the ANSI spec (based in large part on the work of Paul Dietz, a user of Allegro CL who has been developing an extensive test suite for ANSI Common Lisp). There have been a few additional changes in 8.0, though none of significant user-visibility. Users interested in this project should see *Tightening of ANSI Conformance in Allegro CL* and its subsections in *version-70-release-notes.htm*.