<table>
<thead>
<tr>
<th>Document Type</th>
<th>LIGO-T01XXXXXX-Z</th>
<th>2003/10/09</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guidelines for installing LAL and LALWrapper for use with LDAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI working group</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Distribution of this draft:*

LSC and LIGO
Contents

1 Introduction 3

2 Installing LAL and LALWrapper for use with LDAS 4
   2.1 Configuring your environment .......................................... 4
   2.2 Getting LAL and LALWrapper from CVS ............................ 5
   2.3 Compiling LAL in production mode .................................. 5
   2.4 Compiling LAL in debug mode ........................................ 5
   2.5 Stowing LAL .............................................................. 6
   2.6 Compiling LALWrapper in production mode ......................... 6
   2.7 Compiling LALWrapper in debug mode .............................. 7
   2.8 Stowing LALWrapper .................................................. 7

3 Tests of LAL and LALWrapper 8
   3.1 Stand-alone tests of LAL .............................................. 8
   3.2 Stand-alone tests of LALWrapper .................................... 8
   3.3 Test of these build instructions .................................... 8
   3.4 CVS tag name and release number convensions ................... 8

A Understanding LAL/LALWrapper configure arguments 10
1 Introduction

This document is for maintainers of LAL, LALWrapper, or LDAS on a system. It is assumed that there exists all the necessary tools in /ldcg on the system where LAL and LALWrapper are to be installed.
2 Installing LAL and LALWrapper for use with LDAS

Read this section if you are installing a tagged version of LAL and LALWrapper on a production system for running under LDAS.

We describe the installation of LAL and LALWrapper for running in production mode under LDAS. LAL and LALWrapper should be tested in stand-alone mode before production runs. We describe the standard stand-alone tests later. It is assumed that there exists all the necessary tools in /ldcg on the system where LAL and LALWrapper are to be installed.

2.1 Configuring your environment

By the following commands, set the environment variables LALTAG and LALWRAPPERTAG to be the most up-to-date CVS version tags of LAL and LALWrapper respectively. The commands should be modified appropriately. If you are using bash, execute the following lines:

```
LALTAG=HEAD # Change to the most up-to-date version tag!
LALWRAPPERTAG=HEAD # Change to the most up-to-date version tag!
export LALTAG LALWRAPPERTAG
```

If you are using csh or tcsh, execute the following lines:

```
setenv LALTAG HEAD # Change to the most up-to-date version tag!
setenv LALWRAPPERTAG HEAD # Change to the most up-to-date version tag!
```

Typically you will replace HEAD with the most recent release tag, which may look like release-1-0. The tag used for LAL may be different from the tag used for LALWrapper.

In addition, set the environment variable PKGPREFIX to be the directory where you install software packages and the variable BUILD_DIR to the directory where you want to download and build the sources (this may be a temporary directory). For example, one may have PKGPREFIX=/opt and BUILD_DIR=/tmp. For LDAS systems, the following is used. If you are using bash, execute the following lines:

```
PKGPREFIX=/export/ldcg_server/$OSTYPE # Change to where software is installed
LALPREFIX=$PKGPREFIX/lal
STOW_PKGS=$LALPREFIX/stow_pkgs
BUILD_DIR=$STOW_PKGS/src # Change to where build is done
export PKGPREFIX LALPREFIX STOW_PKGS BUILD_DIR
```

If you are using csh or tcsh, execute the following lines:

```
setenv PKGPREFIX /export/ldcg_server/$OSTYPE # Change to where software is installed
setenv LALPREFIX $PKGPREFIX/lal
setenv STOW_PKGS $LALPREFIX/stow_pkgs
setenv BUILD_DIR $STOW_PKGS/src # Change to where build is done
```

Finally, you should make sure that the needed directories exist

```
test -d $PKGPREFIX || mkdir $PKGPREFIX
test -d $LALPREFIX || mkdir $LALPREFIX
test -d $STOW_PKGS || mkdir $STOW_PKGS
test -d $BUILD_DIR || mkdir $BUILD_DIR
```
2.2 Getting LAL and LALWrapper from CVS

Only tagged versions of LAL and LALWrapper should be installed on production systems. The following commands download the tagged version of LAL with CVS tag LALTAG. Change the CVS tag in the environment variable LALTAG to be the most up-to-date version tag.

```
cvs -d:pserver:anonymous@gravity.phys.uwm.edu:/usr/local/cvs/lal login
```
(the password is anonymous)

```
cd $BUILD_DIR

cvs -d:pserver:anonymous@gravity.phys.uwm.edu:/usr/local/cvs/lal \
    checkout -r $LALTAG lal

cd lal
../00boot
```

The last command, `./00boot`, is needed to create the configure script, the makefile inputs, etc.

The following commands download the tagged version of LALWrapper with CVS tag LALWRAPPERTAG. Change the CVS tag in the environment variable LALWRAPPERTAG to be the most up-to-date version tag.

```
cvs -d:pserver:anonymous@gravity.phys.uwm.edu:/usr/local/cvs/lalwrapper login
```
(the password is lalwrapper)

```
cd $BUILD_DIR

cvs -d:pserver:anonymous@gravity.phys.uwm.edu:/usr/local/cvs/lalwrapper \
    checkout -r $LALWRAPPERTAG lalwrapper

cd lalwrapper
./00boot
```

The last command, `./00boot`, is needed to create the configure script, the makefile inputs, etc.

2.3 Compiling LAL in production mode

The LAL software suite should be built in both production and debug modes. To build LAL in production mode, issue the commands:

```
cd $BUILD_DIR/lal

test -d `uname`-`uname -m`-production || mkdir `uname`-`uname -m`-production

cd `uname`-`uname -m`-production

./configure --prefix=$LALPREFIX --disable-debug --enable-frame --enable-mpi \
    --with-extra-cppflags=-I/ldcg/include \n    --with-extra-cflags="-g -O4 -ansi -fexceptions" \n    --with-extra-ldflags=-L/ldcg/lib \n
make
make check
make dvi
make install prefix=$STOW_PKGS/lal-$LALTAG-production
```

2.4 Compiling LAL in debug mode

To build LAL in production mode, issue the commands:
cd $BUILD_DIR/lal

test -d 'uname'-'uname -m'-debug || mkdir 'uname'-'uname -m'-debug

cd 'uname'-'uname -m'-debug

../configure --prefix=$LALPREFIX --enable-debug --enable-frame --enable-mpi \
    --with-extra-cppflags=-I/ldcg/include \
    --with-extra-cflags="-g -ansi -fexceptions" \
    --with-extra-ldflags=-L/ldcg/lib \
make
make check
make dvi
make install prefix=$STOW_PKGS/lal-$LALTAG-debug

2.5 Stowing LAL

To stow the production version of LAL, issue the commands

cd $STOW_PKGS
stow lal-$LALTAG-production

If there is already a version of LAL that is stowed, you will have to delete it first using the stow --delete command.

To change from the production version of LAL to the debug version of LAL, issue the commands

cd $STOW_PKGS
stow --delete lal-$LALTAG-production
stow lal-$LALTAG-debug

and to do the reverse, issue the commands

cd $STOW_PKGS
stow --delete lal-$LALTAG-debug
stow lal-$LALTAG-production

2.6 Compiling LALWrapper in production mode

The LALWrapper software suite should be built in both production and debug modes. To build LALWrapper in production mode, first make sure that the production version of LAL has been stowed (see section 2.5) and then issue the commands:

cd $BUILD_DIR/lalwrapper

test -f Makefile & & make distclean

./configure --prefix=$LALPREFIX --disable-debug \
    --with-extra-cppflags="-I$LALPREFIX/include -I/ldcg/include" \
    --with-extra-cflags="-g -O4" \
    --with-extra-ldflags="-L$LALPREFIX/lib -L/ldcg/lib" \
make
make check
make dvi
make install prefix=$STOW_PKGS/lalwrapper-$LALWRAPPERTAG-production
2.7 Compiling LALWrapper in debug mode

To build LALWrapper in debug mode, first make sure that the debug version of LAL has been stowed (see section 2.5) and then issue the commands:

```
cd $BUILD_DIR/lalwrapper
test -f Makefile && make distclean
./configure --prefix=$LALPREFIX --enable-debug \
  --with-extra-cppflags="-I$LALPREFIX/include -I/ldcg/include" \
  --with-extra-cflags=-g \
  --with-extra-ldflags="-L$LALPREFIX/lib -L/ldcg/lib" \
make
make check
make dvi
make install prefix=$STOW_PKGS/lalwrapper-$LALWRAPPERTAG-debug
```

2.8 Stowing LALWrapper

To stow the production version of LALWrapper, issue the commands

```
cd $STOW_PKGS
stow lalwrapper-$LALWRAPPERTAG-production
```

If there is already a version of LALWrapper that is stowed, you will have to delete it first using the `stow --delete` command. You must also stow the production version of LAL (see section 2.5).

To change from the production version of LAL/LALWrapper to the debug version of LAL/LALWrapper, issue the commands

```
cd $STOW_PKGS
stow --delete lal-$LALTAG-production
stow --delete lalwrapper-$LALWRAPPERTAG-production
stow lal-$LALTAG-debug
stow lalwrapper-$LALWRAPPERTAG-debug
```

and to do the reverse, issue the commands

```
cd $STOW_PKGS
stow --delete lal-$LALTAG-debug
stow --delete lalwrapper-$LALWRAPPERTAG-debug
stow lal-$LALTAG-production
stow lalwrapper-$LALWRAPPERTAG-production
```
3 Tests of LAL and LALWrapper

Read this section if you are maintaining LAL or LALWrapper and you are going to create a tagged version LAL/LALWrapper for installation on a production system.

Before LAL/LALWrapper are installed on a production system, there should be a CVS-tagged version of these packages. (Only tagged versions of LAL and LALWrapper should be used on production systems.) To ensure that LAL and LALWrapper are suitable for installation on a production system, and thus are ready to be tagged, the minimal tests in this section must be performed.

3.1 Stand-alone tests of LAL

LAL contains a script that runs a test suite called testscript.sh. This script should always be run before producing a production release of LAL. The script can take various options; invoke it using .//testscript.sh --help to get a listing of these options. LAL should pass the following test as a minimum:

```bash
./testscript.sh --extra-config-args="--with-extra-ldflags=-L/ldcg/lib\ --with-extra-cppflags=-I/ldcg/include"
```

Each of these tests will take around ten minutes. Always perform these tests before producing a tagged version of LAL.

3.2 Stand-alone tests of LALWrapper

These tests are to be done on a stand-alone version of LAL/LALWrapper using a stand-alone version of wrapperAPI. Follow the instructions in the “How to develop search code to run in LDAS under wrapperAPI” in the LALWrapper Software Documentation (http://www.lsc-group.phys.uwm.edu/lal/lalwrapper.pdf) for getting and installing stand-alone versions of LAL, LALWrapper, and wrapperAPI.

Each shared object to be used with LDAS should have a schema file in the example directory of LALWrapper. These schema files should be executed using mpirun to verify that they run correctly with wrapperAPI.

3.3 Test of these build instructions

Finally, it is important to make sure that the build instructions given in Sec. 2 work. Follow these installation instructions (use HEAD for the CVS tags as you will not have tagged LAL/LALWrapper yet) to make sure that there are no build problems. This test should be done both on Linux-ix86 and SunOS-sparc systems as these are the systems that will be used with LDAS.

3.4 CVS tag name and release number conventions

CVS tags for testing purposes generally have a name of the form test-{date} where ⟨date⟩ is the date in the format returned by the command date --iso-8601. For example: test-2002-06-17. If subsequent tags must be made on the same day, append the tag with a letter, e.g., test-2002-06-17a would be the next tag on that day, test-2002-06-17b the one after that, etc.

CVS tags for releases should be of the form release-{⟨major⟩}-{⟨minor⟩} where ⟨major⟩ is the major release number and ⟨minor⟩ is the minor release number. A released tarball of LAL and LALWrapper is to be made from every release CVS tag, and the LAL webpage will be updated to contain this tarball as
the current release, and the links to the documentation will be updated to point to the documentation in this
release.

Before producing a released version of LAL, the testscript tests described in Sec. 3.1 should be per-
formed on several additional systems in use in the LSC: 64-bit architecture DEC/Alpha (which has a float-
ing point unit that can trap floating point exceptions), SGI/Irix, and Sun/Solaris both with gcc and with the
native cc (usually /opt/SUNWspro/bin/cc).
A Understanding LAL/LALWrapper configure arguments

We detail the meaning of some of the more useful arguments to the configure script for LAL and LALWrapper. For a full list of configure options, execute `configure --help`.

The following configure options are available to both LAL and LALWrapper:

--help Prints a list and short summary of all configure options.

--prefix Prefix directory where the contents of LAL and LALWrapper will be installed. For example, if you use `--prefix=/usr/local` then the libraries will be installed in `/usr/local/lib`, headers in `/usr/local/include` etc. The prefix is used when compiling the code as where everything thinks it will be installed. For example, all rpaths in shared libraries are set up so that thing will work when installed in the specified prefix. However, the prefix can be overridden during the actual installation by resetting the value of prefix in the Makefile. For example, if you want to install in the directory `/usr/local/stow_pkgs` and then later stow so that everything is in `/usr/local`, you `configure` with prefix set to `/usr/local` but when you `install` you issue the command `make install prefix=/usr/local/stow_pkgs`. Note that the library rpaths are set up for `/usr/local`, not `/usr/local/stow_pkgs`, so you will need to stow before the libraries can be used.

--with-cc The C compiler to use. For example, to use the native Solaris C compiler rather than gcc, then you need to use `--with-cc=/opt/SUNWspro/bin/cc`.

--with-extra-cppflags Extra C preprocessor commands that may be needed. These are the `-I` and `-D` options to the compile commands. For example, if you want the compiler to find headers in `/ldcg/include`, then you need to use `--with-extra-cppflags=-I/ldcg/include`. These flags may also be specified in the environment variable CPPFLAGS.

--with-extra-libs Extra libraries and library paths that may be needed for linking. These are the `-L` and `-l` options to the linking commands. For example, if you want the compiler to find libraries in `/ldcg/lib`, then you need to use `--with-extra-libs=-L/ldcg/lib`. These flags may also be specified in the environment variable LIBS.

--with-extra-ldflags Extra linker flags that may be needed, as arguments to the C compiler. For example, if you want to strip symbolic information from the linker output, you need to use `--with-extra-ldflags=-Wl,-s`. These flags may also be specified in the environment variable LD_FLAGS.

--with-extra-cflags Extra compiler flags that may be needed. For example, if you want to include debugger information, you need to use `--with-extra-cflags=-g`. These flags may also be specified in the environment variable CFLAGS.

--enable-shared, --disable-shared Enable/disable the production of shared libraries.

--enable-static, --disable-static Enable/disable the production of static libraries.

--enable-debug, --disable-debug Enable/disable the production of debugging code in LAL and LALWrapper. When debugging is turned on, the LAL library contains a symbol `lalDebugLevel` which determines the level of debugging information that is generated. Furthermore, when compiled with debugging, there is additional code generated to perform debugging diagnostics. When compiled without debugging, this code is removed. For example, all `ASSERT` statements are removed and `LALMalloc`, `LALCalloc`, etc become `malloc`, `calloc`, etc. This means that there will be considerable savings in code speed and memory usage.
In addition, LAL has the additional configure options:

`--enable-mpi, --disable-mpi` Enable/disable the inclusion of routines for performing MPI communication within LAL. This option should be enabled when LAL is being compiled for use with LALWrapper.

`--enable-frame, --disable-frame` Enable/disable the inclusion of routines for performing Frame data format I/O routines within LAL. If enabled, these routines are installed in a separate library, `liblalframe`, which is not available to LALWrapper. It does not matter, from the viewpoint of LALWrapper, whether LAL was configured with `--enable-frame` or not.