ecCodes installation

Overview

ecCodes uses **CMake** for compilation and installation. This is a first step towards an homogenisation of the installation procedures for all ECMWF packages.

Like autotools, CMake will run some tests on the user's system to find out if required third-party software libraries are available and note their locations (paths). Based on this information it will produce the Makefiles needed to compile and install ecCodes.

CMake is a cross-platform free software program for managing the build process of software using a compiler-independent method.

Generating the Makefiles with CMake

One nice and highly recommended feature of CMake is the ability to do **out of source** builds. In this way you can make all your ".o" files, various temporary depend files, and even the binary executables without cluttering up your source tree. To use out of source builds, first create a build directory, then change into your build directory and run cmake pointing it to the source directory and using your own options.

The command gives feedback on what requirements are fulfilled and what software is still required. The following table gives an overview of the different options. The default (without any options) will compile a shared library only and install it in /usr/local/.
Note: When an option is ON by default, it is enabled even if not explicitly requested. Features with default OFF need to be explicitly enabled by the user with -DENABLE_<FEATURE>=ON

cmake options	doc	default
CMAKE_INSTALL_PREFIX	where you want to install your ecCodes	/usr/local
CMAKE_BUILD_TYPE	to select the type of compilation: • Debug • RelWithDebInfo • Release	RelWithDebInfo
BUILD_SHARED_LIBS	Select the type of library built: • ON (Build shared libraries only) • OFF (Build static libraries only) • BOTH (Build both shared and static libraries)	ON
CMAKE_C_COMPILER	C Compiler	
CMAKE_C_FLAGS	Flags for the C Compiler	
CMAKE_Fortran_COMPILER	Fortran Compiler	
CMAKE_Fortran_FLAGS	Flags for the Fortran Compiler	
ENABLE_NETCDF	For the grib_to_netcdf convert tool	ON

ENABLE_JPG	Enable JPEG2000 support. This option should look for Jasper or OpenJPG. If this option is enabled it is also possible to pick which one of OpenJpeg or Jasper to use
	`-DENABLE_JPG_LIBOPENJPEG=ON`
	`-DENABLE_JPG_LIBJASPER=ON`
		ON

ENABLE_PNG	Enable PNG support for decoding/encoding	OFF
ENABLE_AEC	Enable Adaptive Entropy Coding for decoding/encoding (CCSDS)	OFF
ENABLE_PYTHON2	Offers the Python interface to the package. Note: This option is only for **Python 2** (now deprecated) For **Python 3** use "pip3 install". See below	OFF
ENABLE_FORTRAN	Offers the Fortran interface to the package	ON
ENABLE_ECCODES_THREADS	Enable POSIX threads	OFF

ENABLE_ECCODES_OMP_THREADS	Enable OpenMP threads	OFF
ENABLE_MEMFS	See Memory based access to definition/sample files	OFF
ENABLE_EXTRA_TESTS	Enable extended regression testing (which requires data downloads)	OFF

Note: The compilers can also be overridden by setting the environment variables CC and FC.

Note: To see the full output from the compilation, you can use:

```
make VERBOSE=1
```

Quick installation guide

Here is an example of a list of commands you could use to install ecCodes. It is assumed ">" is the shell prompt.

```
> tar -xzf  eccodes-x.y.z-Source.tar.gz
> mkdir build ; cd build

> cmake -DCMAKE_INSTALL_PREFIX=/path/to/where/you/install/eccodes ../eccodes-x.y.z-Source
...

> make
> ctest
> make install
```

It is recommended that you always build in a clean directory and also install into a clean directory.

By default the ctest step above (running the tests) does NOT require any data to be downloaded and only runs basic sanity tests. However if you wish to exercise more of the functionality of ecCodes, you are advised to configure the build with:

```
> cmake ...  -DENABLE_EXTRA_TESTS=ON
```

In this mode the tests will take longer and you need to be connected to the internet (and set the relevant "http_proxy" environment variable if you must use a proxy) so data files can be downloaded for the tests. You can however download all the data files in one go (See the data tarball link at the top of the Releases page)

Also note that if you have several CPUs, you can invoke the "make" and "ctest" commands above in parallel (e.g. "`make -j4`", "`ctest -j4`"). This will speed up the whole build/test process significantly.

Python 3 bindings

The Python 3 bindings are now built with CFFI and are packaged separately. So first install ecCodes as above with a shared library (either BUILD_SHARED_LIBS=ON or BOTH) and then install from PyPI:

```
> pip3 install eccodes
```

or

```
> pip3 install --install-option="--prefix=/path/to/where/you/install/eccodes" eccodes
```

If you are installing into a non-standard installation location (as in the 2nd case above), then set the environment variable ECCODES_DIR to tell the bindings where to find the root installation directory of the ecCodes engine.

```
> export ECCODES_DIR=/path/to/where/you/install/eccodes
```

See: https://pypi.org/project/eccodes/

Microsoft Windows

Please note: Windows is not a platform that is used for technical work at ECMWF. *Therefore we will not be able to support this platform as well as we do for others, such as Linux*. For operational use, we strongly advise you to use Linux.

If you find problems, we invite the Windows user community to use our new set-up on GitHub to fork/contribute and issue pull request changes. Old Reliable Tech has set up automatic tests and builds for Windows on GitHub, which should ensure continued releases for this new platform.

Legacy build system ("autotools")

From ecCodes version 2.0.0 the "autotools" build system is deprecated and no longer supported.