Installation Guide

Overview

Metview uses **CMake** for its compilation and installation, in line with all other ECMWF packages. Note that there are other ways to install Metview from pre-built binary packages, or from a source bundle that includes many dependencies. See Releases.

CMake installation instructions

The **CMake** build system is used to build ECMWF software. The build process comprises two stages:

1. CMake runs some tests on the system and finds out if required software libraries and headers are available. It uses this information to create native build tools (e.g. Makefiles) for the current platform.
2. The actual build can take place, for example by typing '`make`'.

Prerequisite

To install any ECMWF software package, CMake needs to be installed on your system. On most systems it will be already installed or this can be done through the standard package manager to install software. For further information to install CMake see

http://www.cmake.org/cmake/help/install.html

Directories

During a build with CMake there are three different directories involved: The **source dir**, the **build dir** and the **install dir**.

Directory	Use	Example
Source	Contains the software's source code. This is where a source tarball should be extracted to.	`/tmp/src/sw-package`
Build	Configuration and compiler outputs are generated here, including libraries and executables.	`/tmp/build/sw-package`
Install	Where the software will actually be used from. Installation to this directory is the final stage.	`/usr/local`

Of these, the source and build directories can be anywhere on the system. The installation directory is usually left at its default, which is `/usr/local`. Installing software here ensures that it is automatically available to users. It is possible to specify a different installation directory by adding `-DCMAKE_INSTALL_PREFIX=/path/to/install/dir` to the CMake command line.

ECMWF software does **not** support in-source builds. Therefore the build directory **cannot** be (a subdirectory of) the source directory.

Quick Build Example

Here is an example set of commands to set up and build a software package using default settings. More detail for a customised build is given below.

```
# unpack the source tarball into a temporary directory
mkdir -p /tmp/src
cd /tmp/src
tar xzvf software-version-Source.tar.gz

# configure and build in a separate directory
mkdir -p /tmp/build
cd /tmp/build
cmake /tmp/src/software-version-Source
make
```

On a machine with multiple cores, compilation will be faster by specifying the number of cores to be used simultaneously for the build, for example:

```
make -j8
```

If the `make` command fails, you can get more output by typing:

```
make VERBOSE=1
```

The software distribution will include a small set of tests which can help ensure that the build was successful. To start the tests, type:

```
ctest
```

As before if you have multiple cores, you can run the tests in parallel by:

```
ctest -j8
```


Some projects might not be set up to run tests in parallel. If you experience test failures, run the tests sequentially.

If the tests are successful, you can install the software:

```
make install
```

General CMake options

Various options can be passed to the CMake command. The following table gives an overview of some of the general options that can be used. Options are passed to the `cmake` command by prefixing them with **-D**, for example **-DCMAKE_INSTALL_PREFIX=/path/to/dir**.

CMake Option	Description	Default
`CMAKE_INSTALL_PREFIX`	where to install the software	`/usr/local`
`CMAKE_BUILD_TYPE`	to select the type of compilation: • Debug • RelWithDebInfo • Release • Production	RelWithDebInfo (release with debug info)
`CMAKE_CXX_FLAGS`	Additional flags to pass to the C++ compiler	
`CMAKE_C_FLAGS`	Additional flags to pass to the C compiler	
`CMAKE_Fortran_FLAGS`	Additional flags to pass to the Fortran compiler	

The C, C++ and Fortran compilers are chosen by CMake. This can be overwritten by setting the environment variables CC, CXX and F77, before the call to `cmake`, to set the preferred compiler. Further the variable `CMAKE_CXX_FLAGS` can be used to set compiler flags for optimisation or debugging. For example, using `CMAKE_CXX_FLAGS="-O2 -mtune=native"` sets options for better optimisation.

Finding support libraries

If any support libraries are installed in non-default locations, CMake can be instructed where to find them by one of the following methods. First, the option CMAKE_PREFIX_PATH can be set to a colon-separated list of base directories where the libraries are installed, for example `-DCMAKE_PREFIX_PATH=/path/where/my/sw/is/installed`. CMake will check these directories for any package it requires. This method is therefore useful if many support libraries are installed into the same location.

Troubleshooting

Debugging configure failures

If CMake fails to configure your project, run with debug logging first:

```
cmake -DECBUILD_LOG_LEVEL=DEBUG [...] /path/to/source
```

This will output lots of diagnostic information (in blue) on discovery of dependencies and much more.

Requirements to build Metview

The following table lists the dependencies Metview requires to be built from source. Please note, if you install these package from source you also might have to install the respective "-devel" packages.

Compilers		
C++	http://gcc.gnu.org/	
Fortran	http://gcc.gnu.org/fortran/	
Utilities		
make	http://www.gnu.org/software/make/	
Third party packages **(best installed through system package manager)**		
Qt 5	http://www.qt.io/	if Metview's user interface is required. *Note that on some systems it is also necessary to install the libQtWebKit-devel development package (it may have different names on different systems)*
gdbm	http://www.gnu.org.ua/software/gdbm/	
bash	https://www.gnu.org/software/bash/	
netcdf 4	http://www.unidata.ucar.edu/software/netcdf/	Please note: You also will need to install HDF5 and the legacy C++ interface if you wish to run the Single Column Model from Metview (ECMWF only)
curl		Optional for web services support (WMS, Download module)
bison		
flex		
ECMWF libraries		
ecCodes	ecCodes Home	
magics	Magics	if plotting support is needed. Note that Magics should be configured with the **-DENABLE_METVIEW=ON** option. For a 'pure batch' installation of Metview with no user interface, it is possible to supply Magics with the option **-DENABLE_METVIEW_NO_QT=ON**
odb-api	ODB-API Home	if ODB support needed
emoslib	EMOSLIB	optional and deprecated

CMake options used in Metview

CMake options are passed to the `cmake` command by prefixing them with **-D**, for example **-DENABLE_UI=OFF**.

CMake option	Description	Default
ENABLE_UI	enables the Qt-based user interface	ON
ENABLE_PLOTTING	enables plotting capabilities using Magics	ON
ENABLE_MARS	enables MARS access (not required if using through the Web API)	OFF
MARS_LOCAL_HOME	sets the path to where local MARS is installed	
ENABLE_ODB	enables processing and plotting of ODB data	OFF
ENABLE_MARS_ODB	enables ODB capabilities in MARS client	OFF
ENABLE_USAGE_LOG	enables logging of Metview startup calls	OFF
LOG_DIR	path to where to log the Metview startup calls	
METVIEW_SCRIPT	name of the generated Metview startup script	metview
EXTRA_CONFIG_PATH	path to optional directory containing metview_local* script files	
ENABLE_QT_DEBUG	outputs additional log messages from Qt-based modules	OFF
EXTRA_TITLE	build-specific title to add to the log entries	
ENABLE_INPE	enables INPE modules	ON
Path options - only required when support libraries are not installed in default locations		
CMake Option	**Description**	**Notes**
ECCODES_PATH	path to where ecCodes has been installed	
MAGICS_PATH	path to where Magics has been installed	Only required if plotting is enabled
NETCDF_PATH	path to where netCDF has been installed	
ODB_API_PATH	path to where ODB_API has been installed	Only required if ODB is enabled
ODB_PATH	path to where the original ODB has been installed	Optional if ODB is enabled
EMOS_PATH	path to where Emoslib has been installed	Also set EMOS_LIB_NAME
FDB_PATH	path to where fdb has been installed	Only required if MARS is enabled
FLEXTRA_PATH	path to where the FLEXTRA executable has been installed	See Tutorials for more on FLEXTRA

Notes for installers of Metview 3

If you have installed Metview 3 before, then here are some things to note. Metview 5 does not use directly OpenGL for its on-screen graphics; therefore, it is not necessary to build your own Mesa library anymore.

Metview 5 can be installed side-by-side with an existing Metview 3 installation. However, note that the default startup script will be

```
/usr/local/bin/metview
```

so make sure this will not clash with an existing installation. See the table of CMake options for the flag which will allow you to change this.

FAQ

See also the Installation FAQ.