Stay report :

Externalization of LFI-FA formats from The IAL Code

Realized by : CHIKHI Walid
National Office of Meteorology
ALGERIA

Supervised by : Alexandre MARY ; Reyad EL KHATIB
CNRM / Meteo France
FRANCE

CNRM/GMAP - Toulouse
February 06-24 , 2023

<u>Content:</u>

Introduction :

The ACCORD consortium has played a crucial role in the development and improvement of numerical weather prediction (NWP) models, resulting in increased accuracy. However, the complexity of these models has made maintaining and updating the code-source challenging. The development of the NWP code under the ACCORD consortium has been a massive undertaking, resulting in a code-source that is hundreds of thousands of lines long. The growth of the code-source has resulted in several difficulties, including the difficulty of maintaining and updating it to keep up with advances in technology and meteorological science. For that ,the ACCORD consortium has recognized the importance of externalizing parts of the IAL (Ifs/Arpege/LAM) code-source , to insure such maintainability and facilitate model development.
On the other hand, the need for tools developed under the ACCORD consortium or NWP services, such as EPyGrAM and NWP outputs, from external organisations (such as universities…) has highlighted the importance of externalizing the LFI-FA-LFA formats from the IAL code. As long as The use of these formats is totally dependent on compiling the entire IAL pack, which requires high calculation performance that is not always available and who may be limited to local computers.

This report will summarize the work completed during my stay in Meteo-France (Toulouse), having the following tasks as main objectives :

1. Introduction to the the ACCORD and Meteo-France working flow and environment as a newcomer.
2. Externalize a light library for the management of LFI-FA-LFA formats using cmake.
3. Test the library on different platforms, compilers and options.
4. Integrate the library into the IAL code workflow, in particular with the GMKPACK tool.
5. Test of the new IAL binaries using the DAVAI tool.

*1) **Introduction as a newcomer :***

During the first few days of my stay at Meteo-France, I had the opportunity to receive an introduction to the project and familiarize myself with the working environment. Alexandre Mary and Éric Éscaliére provided a comprehensive overview of the workflow and development process used by the consortium, as well as an explanation of Meteo-France's organizational structure . This introduction was invaluable in helping me to understand how my work would fit into the larger context of the project and the goals we were aiming to reach.

*2) **Generalities about the library :***

Alexandre Mary had already made significant progress on some of the preparatory work. He had mainly performed an initial filtering of the main routines for managing LFI-FA formats from IAL code-source. For this, the beginning of my stay was marked by an understanding of Alex’s modset as well as the library external dependencies.

Although at present we have not yet decided on the name of the package or library created , in this report we will refer to this library as «***LFIFA*** » .

*2.1) **Building tools** :*

For this project , CMake and ecbuild were chosen due to their ability to simplify the creation of build scripts and improve the efficiency of the library generation process. CMake is a widely used open-source tool that enables developers to manage and automate the build process for their software projects. CMake allows developers to write platform-independent build scripts and generate makefiles, which can be used to compile their projects on different platforms. ecbuild, on the other hand, is a tool that provides an additional layer of abstraction on top of CMake, simplifying the creation of CMakeLists.txt files and allowing for the generation of consistent and well-organized build scripts across different projects. By using CMake and ecbuild together the library building process was streamlined and ensured consistency and portability throughout the project.

2.2) Code Architecture

The code architecture of a software library is a critical aspect that affects its maintainability, scalability, and usability. A well-designed architecture can make it easier to develop and extend the library, and ensure its long-term sustainability. In this section, we will explore the code architecture of the library we have created and examine its various components, including the directory structure, modules, and interfaces. Below we illustarted the organization of the library code source :

```
LFIFA/
├──── CMakeLists.txt
├──── notes
├──── README.md
├──── src
|    ├──── CMakeLists.txt
|    |  ├────fa
|    |     ├──── CMakeLists.txt
|    |     ├──── api
|    |    |    |    ├──── ellips64.F90
|    |    |    |    ├──── ellips.F90
|    |    |    |    ├──── faauto.F90
|    |    |    |    ├──── facade.F90
|    |    |    |    ├──── facadi.F90
|    |    |    |    ├──── facage.F90
|    |    |    |    ├──── faccpl.F90
|    |    |    |    ├──── facdec.F90
|    |    |    |    ├──── facgra.F90
|    |    |    |    ├──── facgrm.F90
|    |    |    |    ├──── facies.F90
|    |    |    |    ├──── facil1.F90
|    |    |    |    ├──── facile.F90
|    |    |    |    ├──── facilo.F90
|    |    |    |    ├──── facine.F90
|    |    |    |    ├──── facoch.F90
|    |    |    |    ├──── facodega.F90
|    |    |    |    ├──── facodx.F90
|    |    |    |    ├──── facon1.F90
|    |    |    |    ├──── facond.F90
|    |    |    |    ├──── facono.F90
|    |    |    |    ├──── facsim.F90
|    |    |    |    ├──── factec.F90
|    |    |    |    ├──── factui.F90
|    |    |    |    ├──── factum.F90
|    |    |    |    ├──── fadcpl.F90
|    |    |    |    ├──── fadec1.F90
|    |    |    |    ├──── fadeci.F90
```

```
│   │   │   ├── fadeco.F90
│   │   │   ├── fadecoga.F90
│   │   │   ├── fadecx.F90
│   │   │   ├── fadgra.F90
│   │   │   ├── fadies.F90
│   │   │   ├── fadiex.F90
│   │   │   ├── fadoco.F90
│   │   │   ├── fagiot.F90
│   │   │   ├── fagote.F90
│   │   │   ├── fagribex.F90
│   │   │   ├── fagrtr.F90
│   │   │   ├── fagrtw.F90
│   │   │   ├── faicor.F90
│   │   │   ├── faien1.F90
│   │   │   ├── faienc.F90
│   │   │   ├── faieno.F90
│   │   │   ├── faifla.F90
│   │   │   ├── faigra.F90
│   │   │   ├── fainig.F90
│   │   │   ├── fainoc.F90
│   │   │   ├── faiopt.F90
│   │   │   ├── faipag.F90
│   │   │   ├── faipar.F90
│   │   │   ├── fairme.F90
│   │   │   ├── fairno.F90
│   │   │   ├── fais2f.F90
│   │   │   ├── faisan.F90
│   │   │   ├── faisc1.F90
│   │   │   ├── faisc2.F90
│   │   │   ├── faitou.F90
│   │   │   ├── faixla.F90
│   │   │   ├── falais.F90
│   │   │   ├── fa_limits.F90
│   │   │   ├── falimu.F90
│   │   │   ├── falsif.F90
│   │   │   ├── famiso.F90
│   │   │   ├── fandai.F90
│   │   │   ├── fandar.F90
│   │   │   ├── fandax.F90
│   │   │   ├── fanerg.F90
│   │   │   ├── fanfan.F90
│   │   │   ├── fanfar.F90
│   │   │   ├── fanime.F90
│   │   │   ├── fanion.F90
│   │   │   ├── fanmsg.F90
│   │   │   ├── fanouv.F90
│   │   │   ├── fanuca.F90
│   │   │   ├── fanumu.F90
```

```
|    |    |    ├──── faprst.F90
|    |    |    ├──── fapula.F90
|    |    |    ├──── faquin.F90
|    |    |    ├──── farcis.F90
|    |    |    ├──── faregi.F90
|    |    |    ├──── faregu.F90
|    |    |    ├──── fareor.F90
|    |    |    ├──── farflu.F90
|    |    |    ├──── farine.F90
|    |    |    ├──── farpar.F90
|    |    |    ├──── fasgra.F90
|    |    |    ├──── fatale.F90
|    |    |    ├──── fatcha.F90
|    |    |    ├──── fatran.F90
|    |    |    ├──── fautif.F90
|    |    |    ├──── faveur.F90
|    |    |    ├──── favori.F90
|    |    |    ├──── faxion.F90
|    |    |    ├──── iswap8.c
|    |    |    ├──── internal
|    |    |    |    ├──── facilo64.h
|    |    |    |    ├──── facilo_mt64.h
|    |    |    |    ├──── facom2.ixnvms.h
|    |    |    |    ├──── facom2.llmoer.h
|    |    |    |    ├──── facono64.h
|    |    |    |    ├──── facono_mt64.h
|    |    |    |    ├──── fadoco64.h
|    |    |    |    ├──── fadoco_mt64.h
|    |    |    |    ├──── fadoco_mt.h
|    |    |    |    ├──── fagribex.h
|    |    |    |    ├──── faieno64.h
|    |    |    |    └──── faieno_mt64.h
|    |    ├──── fi_libc
|    |    |    ├──── fi_gettimeofday.F90
|    |    |    ├──── fi_libc.c
|    |    |    └──── internal
|    |    |         ├──── cargs.h
|    |    |         └──── fi_libc.h
|    |    ├──── grib_mf
|    |    |    ├──── codega.F90
|    |    |    ├──── confi.F90
|    |    |    ├──── confp_mf.F90
|    |    |    ├──── decfp_mf.F90
|    |    |    ├──── decoga.F90
|    |    |    ├──── gbyte_mf.F90
|    |    |    ├──── gbytes_mf.F90
|    |    |    ├──── gsbite_mf.F90
|    |    |    ├──── gsbyte_mf.F90
```

```
│   │   │   ├──── mxmn_mf.F90
│   │   │   ├──── offset_mf.F90
│   │   │   ├──── packgb.F90
│   │   │   ├──── prtbin_mf.F90
│   │   │   ├──── sbyte_mf.F90
│   │   │   ├──── sbytes_mf.F90
│   │   │   └──── unpagb.F90
│   │   ├──── includes
│   │   │   ├──── ellips.h
│   │   │   ├──── facilo.h
│   │   │   ├──── facilo_mt.h
│   │   │   ├──── facono.h
│   │   │   ├──── facono_mt.h
│   │   │   ├──── fadoco.h
│   │   │   ├──── faieno.h
│   │   │   ├──── faieno_mt.h
│   │   │   └──── falgra.h
│   │   ├──── lfa
│   │   │   ├──── internal
│   │   │   │   └──── lfatail.h
│   │   │   ├──── LFA4py.F90
│   │   │   └──── lfa_R8I4.F90
│   │   ├──── module
│   │   │   ├──── fadup_mod.F90
│   │   │   ├──── fa_mod.F90
│   │   │   └──── grib_api_interface.F90
│   │   ├──── python
│   │   │   ├──── bitbuff.c
│   │   │   ├──── compress.F90
│   │   │   ├──── decompress.F90
│   │   │   ├──── FA4py.F90
│   │   │   ├──── ieee754.h
│   │   │   ├──── ieee_is_nan.c
│   │   │   ├──── init_gfortran.c
│   │   │   ├──── invertcol.F90
│   │   │   ├──── LFI4py.F90
│   │   │   ├──── modd_comppar.F90
│   │   │   ├──── mode_searchgrp.F90
│   │   │   └──── nearestpow2.c
│   │   └──── util
│   │       ├──── decf10.F90
│   │       ├──── grib_get_api_version.c
│   │       ├──── ismax_1.F90
│   │       └──── ismin_1.F90
│   └──── lfi
│       ├──── CMakeLists.txt
│       ├──── lfi_alt
│       │    ├──── lfi_abor.c
│       │    ├──── lfi_altm.c
│       │    ├──── lfi_alts.c
│       │    ├──── lfi_dumm.c
│       │    ├──── lfi_fmul.c
│       │    ├──── lfi_grok.c
│       │    ├──── lfi_hndl.c
│       │    ├──── lfi_intf.c
│       │    ├──── lfi_mess.F90
│       │    ├──── lfi_miss.c
│       │    ├──── lfi_util.c
│       │    ├──── lfi_verb.c
│       │    └──── sdl_srlabort.F90
│       │      ├──── internal
│       │    │    ├──── lfi_abor.h
│       │    │    ├──── lfi_altm.h
│       │    │    ├──── lfi_alts.h
│       │    │    ├──── lfi_args.h
│       │    │    ├──── lfi_call.h
│       │    │    ├──── lfi_dumm.h
│       │    │    ├──── lfi_fmul.h
│       │    │    ├──── lfi_fort.h
│       │    │    ├──── lfi_grok.h
│       │    │    ├──── lfi_hndl.h
│       │    │    ├──── lfi_misc.h
│       │    │    ├──── lfi_miss.h
│       │    │    ├──── lfi_type.h
│       │    │    ├──── lfi_util.h
│       │    │    └──── lfi_verb.h
│       ├──── lfi_hist
│       │    ├──── lfiafm.F90
│       │    ├──── lficap.F90
│       │    ├──── lficas.F90
│       │    ├──── lficfg.F90
│       │    ├──── lficom2.ixc.h
│       │    ├──── lficom2.ixm.h
│       │    ├──── lficom2.ixnims.h
│       │    ├──── lficom2.ixt.h
│       │    ├──── lficom2.llmoer.h
│       │    ├──── lfideb.F90
│       │    ├──── lfiecr.F90
│       │    ├──── lfierf.F90
│       │    ├──── lfifer.F90
│       │    ├──── lfifmd.F90
│       │    ├──── lfifra.F90
│       │    ├──── lfiini.F90
│       │    ├──── ...
│       ├──── module
```

```
|       |    ├──── lfimod.F90
|       |    └──── lfi_precision.F90
|         └──── python
|          ├──── bitbuff.c
|          ├──── compress.F90
|          ├──── decompress.F90
|          ├──── ieee754.h
|          ├──── ieee_is_nan.c
|          ├──── init_gfortran.c
|          ├──── invertcol.F90
|          ├──── LFI4py.F90
|          ├──── modd_comppar.F90
|          ├──── mode_searchgrp.F90
|          └──── nearestpow2.c
├──── cmake
|    ├──── FALFI_Flags.cmake
|    └──── project_summary.cmake
|    └──── Findeccodes.cmake
└──── VERSION
```

3) *Installation of LFIFA :*

The LFIFA library has a number of external dependencies naming : gribex , eccodes and fiat libraries. To make sure that LFIFA will compile and work properly, these libraries and packages must be installed on the system and their associated environment variables must be defined.

If the required libraries and packages are not located in the default system library paths such as ${LD_LIBRARY_PATH} or « /usr/local/lib » , the environment variables eccodes_ROOT, gribex_ROOT, and fiat_ROOT must be defined to specify the location of the corresponding libraries or packages installation directories. Without these libraries, the LFIFA library would not be able to work properly and errors would occur when trying to compile or run LFIFA-dependent programs.

3.1) Compilation using Intel compilers :

The compilation of the first version of the library was completed by generating two initial libraries: ***liblfi.a*** (LFI format only) and ***libfa.so/.a*** (FA-LFI-LFA format) on Belenos using :

- Intel/2018.5.274 compiler (ifort ; icc)
- eccodes 2.20.0 version
- fiat : cloned from my fiat repository forked from ACCORD-NWP ; branch : main (https://github.com/walidchikhi/fiat.git)

```
cmake -S <fiat_src> -B < build_dir>  -DCMAKE_Fortran_COMPILER=ifort  -DCMAKE_C_COMPILER=icc
```

Here below the structure of the installation directory of LFIFA and :

```
LFIFA_install :
├──── include
|    ├──── ellips.h
|    ├──── facilo.h
|    ├──── facilo_mt.h
|    ├──── facono.h
|    ├──── facono_mt.h
|    ├──── fadoco.h
|    ├──── faieno.h
|    ├──── faieno_mt.h
|    ├──── falgra.h
|    └──── lfi_type.h
├──── lib
|    ├──── libfa.so (libfa.a)
|    └──── liblfi.a
├──── module
    ├──── fadup_mod.mod
    ├──── fa_mod.mod
    ├──── lfimod.mod
    ├──── modd_comppar.mod
    └──── mode_searchgrp.mod
```

Firstly, in order to test the generated library, we decided to use it with EPyGrAM. This allowed to easily integrate the LFIFA library into our existing workflow and test its functionality on sample data.

To test LFIFA, we had to copy the generated library libfa.so since it contains the FA - LFI - LFA formats into : **${EPyGrAM_dir}/site/arpifs/libs4py.so** . The first tests on no spectral fields in Belenos showed that EpyGrAM reads and writes both FA and LFI formats correctly.

3.2) Compilation using GNU compilers :

Once LFIFA was compiled on Belenos and worked correctly with EPyGrAM, the next step was to ensure the library was compatible with the GNU compilers which are default compilers on most Linux systems.

```
cmake -S <fiat_src> -B < build_dir>  -DCMAKE_Fortran_COMPILER=gfortran  -DCMAKE_C_COMPILER=gcc
```

This was an important step as it ensured that the library could be used on a wider range of systems. For that 2 compiler versions and 2 Ubuntu release versions was tested. Below the summary of the test done and the results found :

Tab 01 : summary of tests using GNU compilers on different Ubuntu release versions.

	Ubuntu 20.04	**Ubuntu 22.04**	**Belenos**
GNU 9.4.0	Segmentation fault	Successfully	Successfully
GNU 11.3.0	Segmentation fault	Successfully	/

As shown in the table, errors of the "segmentation fault" type occurred during tests with EpyGrAM. Thanks to some inputs from Alex, theses problems have been solved on some cases by adding some flags, but the problem still persists and remains unanswered on other cases. further investigation should and will be done to diagnose the source of these errors.

3.3)<u>Testing of spectral fields on Epygram :</u>

In EPyGrAM, the management of formats and spectral transforms are both handled by the same library called "libs4py.so", which is generated by GMKPACK. Therefore, when using LFIFA libraries alone, it is not possible to directly read spectral fields from IAL outputs. To separate these handling problem in EPyGrAM, an experimental version of EpyGrAM was prepared :(belenos: /home/gmap/mrpe/chikhiw/repositories/epygram_splited) with some minor changes on the following programs :

~chikhiw/repositories/epygram_splited/site/arpifs4py/__init__.py

~chikhiw/repositories/epygram_splited/site/arpifs4py/wtransforms.py

A library has been prepared by Alex on the coope team server "sxcoope1" (MF machines), which contains only the spectral transforms using GMKPACK. This allows managing the tests of spectral fields separately from the formats as shown in the figure below :

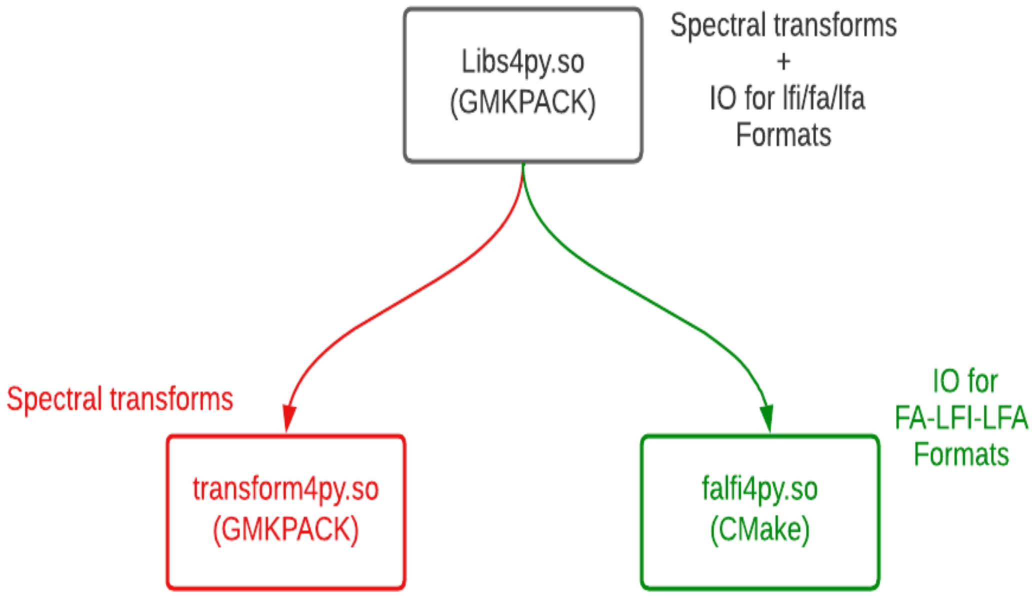

Fig : Separation process of managing spectral transformation and formats in EPyGrAM

Tests for reading and writing spectral fields from an "FA" file have been successfully made. Below is an example of reading and plotting a temperature field on a pressure level:

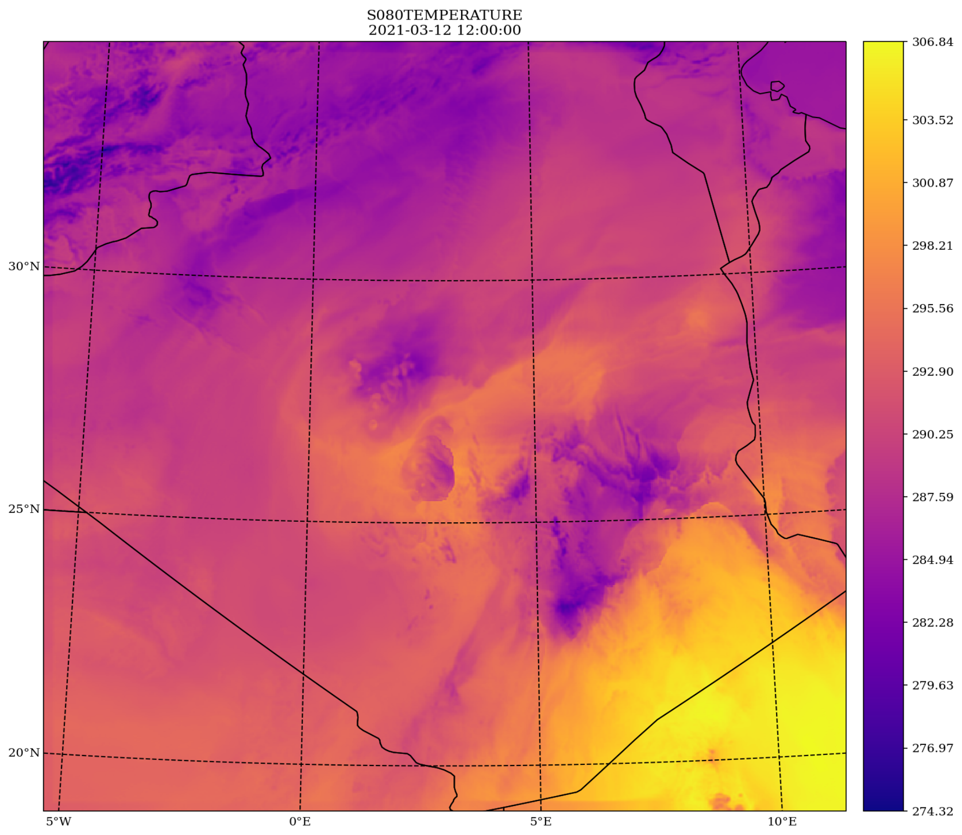

Fig 01 : spectral field test

4) Compiling LFIFA With GMKPACK :

With the movement towards externalization, GMKPACK still manages the compilation of the externalized libraries and packages such as Fiat, trans, and LFIFA. This allows for more efficient management of dependencies and helps to ensure that each package is built correctly.

After generating the pack, blocks below are added to the configuration file in order to integrate LFIFA into the entire IAL compiling process respecting the chronological order for the compilation of external libraries.

Entire packs and files could be found under the path tree or in github :

LFIFA (PRIVATE)	**github**	walidchikhi/LFIFA --branch master
	Hendrix	~chikhiw/STAGE_2022/src/LFIFA.tar.gz
IAL pack	**github**	ACCORD-NWP/IAL --branch mary_CY48T1_preT2
	belenos	~chikhiw/repositories/IAL
GMKPACK config file	**belenos**	*~chikhiw/SAVE/STAGE_2022/gmkfile.x*

Below the block added to The gmkpack configuration file :

```
# List of projects in hub (ordered sort to enable dependencies)
GMK_HUB_PROJECTS = ecSDK OOPS Atlas Fiat Ectrans Lfifa

GMK_HUB_LIBRARIES_IN_Lfifa = lfifa
GMK_HUB_METHOD_FOR_Lfifa = cmake

GMK_CMAKE_lfifa = -Wno-deprecated -Wno-dev -DCMAKE_C_COMPILER=\${VCCNAME} -
DCMAKE_C_FLAGS=\"${VCCFLAGS} ${OPT_VCCFLAGS}\" -DCMAKE_CXX_COMPILER=\$
{CXXNAME} -DCMAKE_CXX_FLAGS=\"${VCCFLAGS} ${OPT_VCCFLAGS} ${MACROS_CXX}\" -
DCMAKE_Fortran_COMPILER=\${FRTNAME} -DCMAKE_Fortran_FLAGS=\"${FRTFLAGS} $
{OPT_FRTFLAGS} ${MACROS_FRT}\" -DCMAKE_BUILD_TYPE=NONE  -Decbuild_ROOT=\$
{TARGET_PACK}/\${GMK_HUB_DIR}/\${GMK_LAST_HUB_BRANCH}/\${GMKSRC}/ecSDK -
Dfiat_ROOT=\${TARGET_PACK}/\${GMK_HUB_DIR}/\${GMK_LAST_HUB_BRANCH}/\$
{GMK_HUB_INSTALL}/Fiat -Dgribex_ROOT=\${AUX_INSTALLDIR} -DBUILD_SHARED_LIBS=BOTH
-DENABLE_SINGLE_PRECISION=OFF   -DENABLE_GPU=OFF -Deccodes_ROOT=\$
{ECCODES_INSTALLDIR}
# lfifa
LD_USR_FA = fa
LD_USR_LFI = lfi
```

Besides the changes made to the configuration file, the LFIFA libraries must be linked to the MASTERODB binary , for this we have added the following block in the link and dependency management file of GMKPACK under : $HOME/***.gmkpack/link/hub_libs*** **.** as static links :

```
LNK_STATIC
.....
LD_USR_FA
LD_USR_LFI
LNK_DYNAMIC
```

5) ***LFIFA routines used by other projects:***

The next two subroutines are among the outsourcing problems encountered during this externalization of these formats :

- ifsaux/module/grib_api_interface.F90.
- ifsaux/utility/iswap8.c

It is important to note that these routines are also used by other programs under the IAL project. Thus, duplicating them would cause conflicts during the generation of binaries such as MASTERODB. It is therefore crucial to consider these implications before deciding to duplicate routines in order to ensure the stability and reliability of the project. For these reasons, since the IAL and LFIFA projects both depend on the "*fiat*" project, these two subroutines have been added in the «fiat» project available on my github ***walidchikhi/fiat*** repository on github under the branch ***fiat_grib_api*** (version based on the branch --mirror-CY48T3_mrg48R1.02 from ACCORD-NWP/fiat) in a way that the subroutine « iswap8.c » is always compiled with fiat but grib_api_interface.F90 and the use of the eccodes package will only be under a certain condition or option that we have defined (-DENABLE_ECCODES=ON/OFF) so that no change on the workflow of the original version is made if this option remain disabled.

5.1) ***Modification in fiat package***

The modifications made under fiat can be summarized in the following points:

1/ fiat/CMakeLists.txt

a. added an option ECCODES

```
ecbuild_add_option(FEATURE ECCODES
                        DEFAULT OFF
                  DESCRIPTION " Compile with ECCODES AND GRIB_API_INTERFACE.F90" )
```

b. Added a block for searching the "eccodes" package and defining the precision -DPARKIND_PRECISION=dp/sp to be chosen to select the parkind_dp/sp modules needed for grib_api_interface.F90.

```
if(HAVE_ECCODES)
   find_package(eccodes REQUIRED)
endif()
```

2) ***fiat/cmake:***

A cmake configuration file named ***Findeccodes.cmake*** has been added to the project under ***fiat/cmake*** which will be used to search for the eccodes package and define the relevant variables.

3) ***fiat/src/CMakeLists.txt***

Permutation in the compilation order between parkind and fiat directories, so that grib_api_interface.F90 finds parkind_${precision} already compiled.

4) fiat/src/fiat/CMakeLists.txt

a. Since the fiat library source code selection method is based on an exhaustive selection of all files with *.F90 and *.c formats, we were forced to remove grib_api_interface.F90 from the source code if the ECCODES option is not enabled.

```
if( NOT ENABLE_ECCODES)
   list(REMOVE_ITEM fiat_src eccode/grib_api_interface.F90 )
endif()
```

b. A new code block has been added to the project that will link fiat to the required eccodes includes and libraries needed to compile grib_api_interface.F90, but only if the ECCODES option is enabled.

```
if(HAVE_ECCODES)
   target_include_directories(fiat PUBLIC ${ECCODES_INCLUDE_DIR})
   target_link_libraries(fiat PUBLIC ${ECCODES})
   target_link_libraries(fiat PUBLIC ${ECCODES_F90})
   target_link_libraries(fiat PUBLIC parkind_${PARKIND_PRECISION})
   target_include_directories(fiat PRIVATE ${CMAKE_SOURCE_DIR}/src/parkind)
endif()
```

5.2) compile fiat with new features :

The options and modifications added under the fiat package are only applicable under certain conditions. For the new compilation of Fiat to be used with LFIFA, it is required to add a specific argument (-DENABLE_ECCODES=ON) to the cmake command while compiling the pack.

The choice of precision in compiling grib_api_interface.F90 is determined by the ENABLE_SINGLE_PRECISION and ENABLE_DOUBLE_PRECISION compiler options. If one of these options is enabled, fiat will compile the module using the corresponding precision. If both options are enabled, fiat will use the double precision option.

***Exemple*:** In this exemple, we will compile fiat with eccodes and grib_api_interface.F90 on double precision :

```
cmake -S <fiat_src> -B < build_dir>  -DENABLE_ECCODES=ON   -ENABLE_SINGLE_PRECISION=OFF
```

These arguments will ensure that the new compilation of fiat is optimized for a version of LFIFA which does nor contain grib_api_interface.F90 neither iswap8.c ; these version of LFIFA could be found on :

Github	**repository**	walidchikhi/LFIFA
	Branch	lfifa_grib_api
Hendrix	~chikhiw/STAGE_2022/LFIFA_grib_api.tar.gz	

6) Summary of paths and github repositories :

Tab 2 : summary of package and pack paths

Package / file	Note	Hendrix path	github		
			repository	branch	security
LFIFA	First version + grib_api_interface.F90 + iswap.c	~chikhiw/ STAGE_2022/ src/LFIFA.tar.gz	walidchikhi/ LFIFA	master	PRIVATE
LFIFA	**Without** grib_api_interface.F90 & iswap.c	~chikhiw/ STAGE_2022/ src/ LFIFA_grib_api.tar.gz	walidchikhi/ LFIFA	lfifa_grib_api	PRIVATE
fiat	original	~chikhiw/ STAGE_2022/ src/fiat.tar.gz	walidchikhi/ **fiat**	main	PUBLIC
fiat	Original + grib_api_interface.F90 + iswap8.c	~chikhiw/ STAGE_2022/ src/ fiat_grib_api.tar.gz	walidchikhi/ **fiat**	fiat_grib_api	PUBLIC
IAL	pre-CY49	**Belenos :** ~chikhiw/repositories/IAL	ACCORD-NWP/IAL	mary_CY48T1_preT2	PRIVATE

Conclusion :

During my stay at Meteo-France, we were able to make significant progress towards the externalization of the FA-LFI formats from the IAL project, as well as developing a lightweight library for their compilation and use. We successfully tested and validated the library using the "EPyGrAM" tool with multiple compiling options.

However, the DAVAI-Test series were not ready to test the generated binaries from IAL such as MASTERODB. Despite this setback, we remain committed to addressing this issue in the future. We will continue collaborating to optimize the code for these formats, making it more user-friendly and accessible to future users. We also aim to complete the DAVAI test series as well as further cleaning up and optimizing the code.

Acknowledgments:

I want to express my gratitude to Alexandre MARY, and Ryad El Khatib for their exceptional support and guidance during my stay. I want also thank all the GMAP staff for the warm reception, especially Claude Fischer, Eric Éscaliére, and Ghislaine Faure.