INSTALLATION GUIDE OF THE MTG FCI L1C DECOMPRESSION SOFTWARE

Doc.No.	:	EUM/MTG/MAN/16/858121
Issue	:	v1A e-signed
Date	:	15 February 2019
WBS/DBS	:	

EUMETSAT
Eumetsat-Allee 1, D-64295 Darmstadt, Germany
Tel: +49 6151 807-7
Fax: +49 6151 807 555
http://www.eumetsat.int

Document Signature Table

	Name	***Function***	***Signature***	***Date***
Prepared by:	Carlos Ferrao	Mission Data Processing Engineer		
Reviewed by:	Fausto Roveda	Data Processing System Facility Manager		
Reviewed by:	Maria Jose Marquez	Quality Assurance Engineer for GEO Programmes		
Reviewed by:	Claude Ledez	Instrument Data Processing Engineer		
Reviewed by:	Claude Keppenne:	Ground Segment Engineering Team Leader		
Approved by:	Susanne Dieterle	Ground Segment Technical Manager		

Distribution List

Name	***Organisation***
As per signature table	EUMETSAT

Document Change Record

Issue / Revision	***Date***	***DCN. No***	***Changed Pages / Paragraphs***
1.0	13-06-2016		First Version
1.1	15-02-2019		Aligned to SIG 1.6 of v1.0.2 Added note in §2.3.2 to indicate that the filter is currently not compatible with HDF1.10.x.

Table of Contents

Table of Figures

1 INTRODUCTION

This document is the Software Installation Guide (SIG) of the MTG FCI L1c decompression software.

1.1 Purpose

The purpose of this document is to provide instructions on how to install the FCI decompression software on end-users systems. The usage of the software is described in the Software User Guide [SUG].

1.2 Scope

This document may interest the users of the FCI L1c decompression software, in particular developers of FCI Test Datasets, the developers and testers of systems which read FCI L1c data, Satellite Application Facilities and End-Users.

1.3 Applicable Documents

None.

1.4 Reference Documents

[SUG]	Software User Guide of the MTG FCI L1c decompression software	EUM/MTG/MAN/16/858121

1.5 Document Structure

This document is organised as follows:

- Section 1 provides the scope and content of the document;
- Section 2 is the FCI L1c Decompression Software Installation Guide:
  - Section 2.1 describes the purpose and the content of the FCI L1c decompression software.
  - Section 2.2 describes the prerequisites for the software installation.
  - Section 2.3 describes the tasks that have to be performed before the installation of the software.
  - Section 2.4 describes the usage of the build and install tools
  - Section 2.5 describes the installation procedure.
  - Section 2.6 describes the update procedure.
  - Section 2.6 describes the procedure for the software removal

2 FCI DECOMPRESSION SOFTWARE INSTALLATION GUIDE

2.1 Software product overview

2.1.1 Purpose of the software

The purpose of the FCI decompression software is to decompress the data that have been compressed with the FCI compression software in netCDF-4 files. These data are namely the effective radiance and pixel quality flags in the FCI L1c datasets.

2.1.2 Content of the software

The FCI decompression software is composed of modules in beige that are shown in Figure 1.
It includes:

- The façade to CharLS (fcicomp-jpegls) which allows separating the core of the FCI decompression software from CharLS processing functions.
- The HDF5 decompression filter (fcicomp-H5Zjpegls) is the filter that is dynamically loaded by HDF5 to decompress the data that have been encoded in JPEG-LS HDF5 compression filter.

Modules in blue are COTS components that have to be installed on the system in order to use the FCI decompression software. They are:

- CharLS software: the optimized implementation of the JPEG-LS standard for lossless and near-lossless image compression;
- The HDF5 library which is the storage layer used by netCDF-4 to read data from disk and write data to disk.
- The standard netCDF-4 C library which is used for accessing datasets from files in the netCDF-4 format.

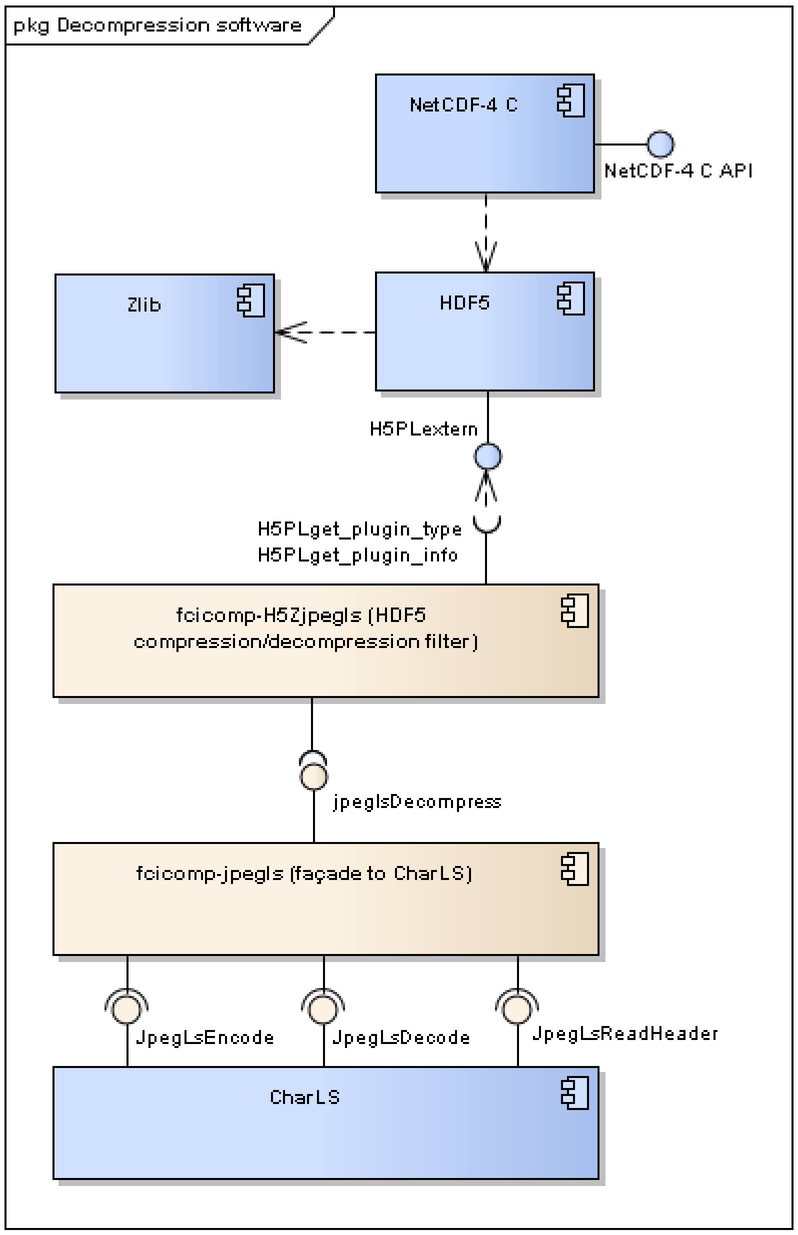

Figure 1 : Modules in the FCI decompression software

2.2 Installation prerequisites

2.2.1 Hardware

The FCI decompression software is known to work properly on the following hardware:

- PC Dell Precision T1650 equipped with 4 Intel Xeon CPU E3-1225 V2 @ 3.2GHz, 8MB and with 3.7GB 1600MHz DDR3 memory running CentOS release 6.3 (64 bits).

The FCI decompression software may work properly on the following hardware configuration:

- DELL Optiplex 9020 Small Form factor models (or equivalent HP model) as described below:
    - INTEL Core i5 (including 4 CPUs with at least 3.2 GHz frequency).
    - 8 GByte DDR3 RAM for all PCs.
    - CentOS 6.5 LINUX distribution

The disk space used after installation is 31MB: less than 1MB for the FCI decompression software and 30MB for the COTS (Zlib, HDF5, CharLS, netCDF-C).

2.2.2 Software

2.2.2.1 OS

The FCI decompression software is known to work properly on the following OS:

- Linux CentOS 6.3 x64 Edition
- Red Hat Enterprise Linux Client release 5.2 (Tikanga)
- Red Hat Enterprise Linux 6.5 Server x64 Edition

2.2.2.2 External libraries and tools required

In order to build and install the FCI decompression software the following software needs to be available on the system:

Tool	Minimum version	Purpose	Website
Gcc	4.4.6	Compiler for the FCI decompression software.	https://gcc.gnu.org
GNU make	3.81	Tool to build and install the FCI decompression software.	www.gnu.org/software/make
CMake	2.8.12	Tool to build the FCI decompression software.	www.cmake.org

The FCI decompression software is also linked to the following libraries that are also required to be installed on the system:

Tool	Minimum version	Purpose	Website
CharLS	1.0	Optimized implementation of the JPEG-LS standard for lossless and near-lossless image compression.	http://charls.codeplex.com/
Zlib	1.2.8	Lossless data-compression library that implement the "deflate" method. This library is required by HDF5.	http://www.zlib.net/

Tool	Minimum version	Purpose	Website
HDF5	1.8.14	Data model, library, and file format for storing and managing data. It is the storage layer used by netCDF-4. **Note**: The HDF5 library should have been configured with the --enable-hl option during its building process.	http://www.hdfgroup.org/HDF5/
netCDF-C	4.3.2	Set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. **Note**: The netCDF-C library should have been configured with the --enable-netcdf-4 option during its building process.	http://www.unidata.ucar.edu/software/netcdf/

The HDF5 library and netCDF-4 libraries should be installed in standard locations on the system (i.e. "/usr/lib" or "/usr/local/lib" on Unix/Linux OS) Otherwise, the installation location shall be known and provided during the build process of the FCI decompression software (See section 2.5.3).

The following COTS are optional:

Tool	Recommended version	Purpose	Website
sha256sum	4.4.6	Compute and check SHA256 message digest. Sha256sum is part of the GNU coreutils package.	https://www.gnu.org/software/coreutils/
libcurl	7.19.7	Client-side URL transfer library. This library is required to build the DAP client of netCDF.	http://curl.haxx.se/

The following software could be used with FCI decompression software (optional):

Tool	Minimum version	Purpose	Website
ncview	2.1.2	Visual browser for netCDF files distributed with a GNU GPL licence.	http://meteora.ucsd.edu/~pierce/ncview_home_page.html

2.2.3 Environment configuration

The FCI decompression software can be installed at any location on the system.
Deploying the FCI decompression software to standard directories for user programs (i.e. in the "/usr/" or "/usr/local/" directories on Unix/Linux OS) requires write permissions to those directories or sudo rights.

2.3 Pre-installation tasks

2.3.1 Building and installation software

It is recommended to install the binary package of gcc, make and cmake coming with the Unix/Linux distribution of the system (e.g. using "yum" under Redhat like Linux OS or "apt" under Debian like Linux OS).
The required version and download links of gcc, make and cmake are provided in section 2.2.2.2.
For information purposes only, the procedure to build the delivered COTS from sources is also given in Annex A.

2.3.2 HDF5

HDF5 library has to be installed on the system. The minimum required version is HDF5 1.8.14.
Warning: The filter is currently not compatible with HDF5 version 1.10.x.

The download link for HDF5 1.8.14 and the recommended installation procedure are given in the Annex A.
Pre-built binary distributions and source code of the current release can be downloaded from:
http://www.hdfgroup.org/HDF5/release/obtain5.html

Note: Zlib 1.1.2 or later is required to build HDF5 1.8.14 from sources even if the FCI decompression software does not use it: HDF5 library dynamically linked to Zlib. Zlib is included in the pre-built binary distributions of HDF5 or can be downloaded from: http://www.zlib.net/. Installation instruction can be found in the README file included in Zlib distribution.

Building HDF5 with configure
Building instructions of the HDF5 current release with configure can be found here:
http://www.hdfgroup.org/HDF5/release/obtainsrc.html#conf
Warning: Make sure to enable the high level library, eventually using the "--enable-hl" option when calling configure. This is required to build netCDF-4.

Building HDF5 with cmake
Building instructions of the HDF5 current release with cmake can be found here:
http://www.hdfgroup.org/HDF5/release/cmakebuild.html
Warning: Make sure to enable the high level library, eventually switching the cmake option HDF5_ENABLE_HL to ON. This is required to build netCDF-4.

2.3.3 netCDF C library

NetCDF C library has to be installed on the system. The minimum required version is netCDF-4.3.2
NetCDF source code can be downloaded from:

http://www.unidata.ucar.edu/downloads/netcdf/index.jsp
Building instructions can be found here:
http://www.unidata.ucar.edu/software/netcdf/docs/getting_and_building_netcdf.html

2.3.4 CharLS

CharLS is required by the FCI decompression software. It has to be installed on the system.
CharLS 1.0 source code can be downloaded from here: http://charls.codeplex.com/
The building procedure is the following:
Unzip CharLS software into the charls directory and move into that directory:

```
unzip CharLS-source-1.0.zip -d charls
cd charls
```

CharLS source code needs to be patched before compiling (see https://charls.codeplex.com/workitem/7823). Patch the source code:

```
sed -i.ori '1s/^/#include "header.h"\n/' defaulttraits.h
```

This add the line “ #include "header.h" ” into the file “defaulttraits.h”.
If sed is not available on the system, this operation can be manually performed editing the file “defaulttraits.h” with a text editor and adding the line “ #include "header.h" at the beginning of this file.
Then, create a building directory and move into it and build CharLS shared library in release mode:

```
mkdir release
cd release
cmake -Dcharls_BUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=RELEASE ..
make
```

This creates the file “libCharLS.so” in the charls/release directory.
CharLS can then be installed using the following commands. Replace the ${CHARLS_INSTALLATION_PATH} by the path where CharLS has to be installed on the system (e.g. “/usr/local/”).

```
cp libCharLS.so ${CHARLS_INSTALLATION_PATH}/lib
cp ../config.h ../interface.h ../publictypes.h \
         ${CHARLS_INSTALLATION_PATH}/include
```

2.4 Usage of the build and install tools

This section describes the usage of the build and installation scripts that can be used to build and install the FCI decompression software modules on the system. The full installation procedure is detailed step by step in section 2.4.3.

- build.sh   Build a module of the FCI decompression software
- install.sh Install a module of the FCI decompression software.

These scripts are located in the folder “gen/” after the software sources are extracted. See section 2.5.2 for more details on the software sources extraction.

2.4.1 Building script

The script “gen/build.sh” is used to build the modules of the FCI decompression software.
It can be launched using the following command:

```
./gen/build.sh module release [cmake-options]
```

Where “*module*” is the FCI decompression software module to build and “cmake-options” is an optional list of options to pass onto the cmake command line and “release” keyword is used to build the software module in release mode as opposed to the debug mode which shall only be used for debugging purposes.
“cmake_option” may include the following options:

```
-DCMAKE_PREFIX_PATH="/path/to/a/cots1/;/path/to/cots2/"
-DCMAKE_INSTALL_PREFIX=/installation/path/
```

CMAKE_PREFIX_PATH has to contain the paths to all the COTS that are required by the module being built. The list of paths must be separated by a semi-colon “;”.
CMAKE_INSTALL_PREFIX is the module installation path. If it is not set, the module will be installed in the default standard location e.g. in the “/usr/lib/” (if the module is a library) or “/usr/local/hdf5/lib/plugin/” (if the module is an HDF5 filter).
The script “./gen/build.sh” creates the “build/*module*” directory at the root of the FCI decompression software source and then builds the module into that directory. The following command is called:

```
cmake [cmake-options] -DCMAKE_BUILD_TYPE=Release module && make
```

The script “gen/build.sh” can also be used to run the unit test of the modules of the FCI decompression software using the following command:

```
./gen/build.sh module test
```

This runs the unit test of the module previously build in the directory “build/*module*”. The following command is called:

```
ctest --output-on-failure
```

2.4.2 Installation script

The script “gen/install.sh” is used to install the modules of the FCI decompression software.
It can be launched using the following command:

```
./gen/install.sh module
```

where “module” is the FCI decompression software module to install.
Use “sudo” to install the module in standard directories for user programs that require sudo privileges (i.e. in the “/usr/” or “/usr/local/” directories on Unix/Linux OS):

```
sudo ./gen/install.sh module
```

The module is installed in the location specified by the -DCMAKE_INSTALL_PREFIX if it has been set at the building time. Otherwise the module is installed in the default location. The following command is called:

```
make install
```

After that command is called, the installation script copies the install_manifest.txt file generated in the building directory to the share/cmake sub-directory in the installation directory.

2.4.3 Logging system

Four different verbosity levels are defined:

ERROR_SEVERITY	Display error message only
WARNING_SEVERITY	Display warnings and error messages
NORMAL_SEVERITY	Display warning, errors and normal messages
DEBUG_SEVERITY	Highest verbosity level for debugging purpose only

The default logging level is the ERROR_SEVERITY.
The verbosity level can be changed at building time using the option:

```
-DLOGGING_LEVEL=severity
```

where *severity* is one of the 4 verbosity levels.
Example: To set the logging level to WARNING_SEVERITY in the fcicomp-jpegls module, use the following command line at building time:

```
./gen/build.sh fcicomp-jpegls release \
         -DLOGGING_LEVEL=WARNING_SEVERITY \
-DCMAKE_PREFIX_PATH=${CHARLS_INSTALLATION_PATH} \
-DCMAKE_INSTALL_PREFIX=${FCIDECOMP_INSTALLATION_PATH}
```

It is possible to disable all the messages (even the error messages) at building time using the option:

```
-DLOGGING=OFF
```

Example: To disable all the messages of the fcicomp-jpegls module, use the following command line at building time:

```
./gen/build.sh fcicomp-jpegls release \
         -DLOGGING=OFF \
-DCMAKE_PREFIX_PATH=${CHARLS_INSTALLATION_PATH} \
-DCMAKE_INSTALL_PREFIX=${FCIDECOMP_INSTALLATION_PATH}
```

2.5 Installation procedure

2.5.1 Software delivery

The FCI decompression software delivery tree is represented below. The variable ${FCIDECOMP_VERSION} is the version of the software, e.g. “V1.0.2”.

```
Delivery
|-- FCIDECOMP-${FCIDECOMP_VERSION}
|    |-- SCF
|    |     |-- FCIDECOMP-SCF.txt                Main SCF file
|    |     `-- other-files.txt                  Other SCF files
|     `-- Software
|          `-- fcidecomp-source-version.tar.gz  Software source code
          archive
`-- sha256_FCIDECOMP_version.txt                SHA256 checksum
```

The “fcidecomp-source-${FCIDECOMP_VERSION}.tar.gz” contains the source code of the FCI decompression software.
Before extracting the software, it is recommended to check the file integrity checking that the SHA256 sums of the files delivered match with the checksums provided in the file checksum.txt.
In the following command lines, replace the ${PATH_TO_DELIVERY} by the delivery path of the FCI decompression software and ${FCIDECOMP_VERSION} by the software version.

```
cd ${PATH_TO_DELIVERY}
sha256sum -c sha256_FCIDECOMP_${FCIDECOMP_VERSION}.txt
```

2.5.2 Software delivery extraction

Extract the source code of the FCI decompression software from the “fcidecomp-source-${FCIDECOMP_VERSION}.tar.gz” archive at the desired location on the system, e.g. into the “/usr/local/src/” directory or into any other writable location on the system.
Replace ${FCIDECOMP_EXTRACTION_PATH} by the path where the source code of the FCI decompression software has to be extracted.

```
mkdir -p ${FCIDECOMP_EXTRACTION_PATH}
tar zxf
         ${PATH_TO_DELIVERY}/FCIDECOMP_${FCIDECOMP_VERSION}/Software/fcide
         comp_sources-${FCIDECOMP_VERSION}.tar.gz --strip 1 -C
         ${FCIDECOMP_EXTRACTION_PATH}
```

This creates the following directory structure:

```
${FCIDECOMP_EXTRACTION_PATH}/
|-- cmake                            Cmake modules and configuration files
|-- fcicomp-common                   Common functions: logging system
|-- fcicomp-jpegls                   Interface to CharLS library
|-- fcicomp-H5Zjpegls                HDF5 JPEG-LS encode/decode filter
|-- fcidecomp-test                   Test script
|-- gen                              Generation scripts
```

```
`-- SCF_fcidecomp                         Software Configuration Files
```

2.5.3 Build and installation of the software modules

Move into the directory where the source code of the FCI decompression software has been extracted.

```
cd ${FCIDECOMP_EXTRACTION_PATH}
```

In the following, all the relative paths are relative to this directory.

2.5.3.1 Build and installation of the fcicomp-jpegls component

Build the façade to CharLS indicating the location of CharLS on the system.
In the following command, replace ${CHARLS_INSTALLATION_PATH} by the installation path of CharLS (e.g. “/usr/local/”). Replace the ${FCIDECOMP_INSTALLATION_PATH} by the path where the module should be deployed. The default installation path is “/usr/”.

```
./gen/build.sh fcicomp-jpegls release \
    -DCMAKE_PREFIX_PATH=${CHARLS_INSTALLATION_PATH} \
    -DCMAKE_INSTALL_PREFIX=${FCIDECOMP_INSTALLATION_PATH}
```

This creates the “build/fcicomp-jpegls” directory.
Run the unit tests:

```
./gen/build.sh fcicomp-jpegls test
```

If all the tests have passed, the module can be installed:

```
./gen/install.sh fcicomp-jpegls
```

or:

```
sudo ./gen/install.sh fcicomp-jpegls
```

This creates the following list of files and symbolic links in the “${FCIDECOMP_INSTALLATION_PATH}/lib” directory:

```
libfcicomp_jpegls.so -> libfcicomp_jpegls.so.1
libfcicomp_jpegls.so.1 -> libfcicomp_jpegls.so.1.0.1
libfcicomp_jpegls.so.1.0.1
```

The “->” represent the symbolic links. The file “libfcicomp_jpegls.so” is a symbolic link pointing on the library “soname” symbolic link “libfcicomp_jpegls.so.1” (API version) and in turn on the library “real name” file “libfcicomp_jpegls.so.1.0.1” (build version).
The file “fcicomp_jpegls.h” is copied into the “${FCIDECOMP_INSTALLATION_PATH}/include” directory.

2.5.3.2 Build and installation of the fcicomp-H5Zjpegls component

Build the HDF5 JPEG-LS decoding filter indicating:

- the installation path of the fcicomp-jpegls module,
- the location of Zlib library,
- the location of HDF5.

In the following command line, make sure to have the double quotes (" ") enclosing the list of CMAKE_PREFIX_PATH paths and no space in between. The CMAKE_PREFIX_PATH should contain the installation path of the fcicomp-jpegls component ${FCIDECOMP_INSTALLATION_PATH}, of Zlib library ${ZLIB_INSTALLATION_PATH} and of HDF5 ${HDF5_INSTALLATION_PATH}. Replace the ${FCIDECOMP_INSTALLATION_PATH} by the path where the module should be deployed. The default installation path is "/usr/local/".

```
./gen/build.sh fcicomp-H5Zjpegls release \
    -DCMAKE_PREFIX_PATH="${FCIDECOMP_INSTALLATION_PATH};
          ${ZLIB_INSTALLATION_PATH};${HDF5_INSTALLATION_PATH}" \
    -DCMAKE_INSTALL_PREFIX=${FCIDECOMP_INSTALLATION_PATH}
```

This creates the "build/fcicomp-H5Zjpegls" directory.
Run the unit tests:

```
./gen/build.sh fcicomp-H5Zjpegls test
```

This installs the file "libH5Zjpegls.so" in the "${FCIDECOMP_INSTALLATION_PATH}/hdf5/lib/plugin" directory and copies the file "H5Zjpegls.h" in the "${FCIDECOMP_INSTALLATION_PATH}/include" directory.
If all the tests have passed, the module can be installed:

```
./gen/install.sh fcicomp-H5Zjpegls
```

This creates the following list of files and symbolic links in the "${FCIDECOMP_INSTALLATION_PATH}/hdf5/lib/plugin" directory:

```
libH5Zjpegls.so -> libH5Zjpegls.so.2
libH5Zjpegls.so.2 -> libH5Zjpegls.so.2.0.0
libH5Zjpegls.so.2.0.0
```

The "->" represent the symbolic links. The file "libH5Zjpegls.so" is a symbolic link pointing on the library "soname" symbolic link "libH5Zjpegls.so.2" (API version) and in turn on the library "real name" file "libH5Zjpegls.so.2.0.0" (build version).
The file "H5Zjpegls.h" is copied into the "${FCIDECOMP_INSTALLATION_PATH}/include" directory.

2.5.3.3 Final installation tree

A possible configuration of the final installation tree is represented below.

```
${FCIDECOMP_INSTALLATION_PATH}/
├── hdf5
│   └── lib
│       └── plugin
│           ├── libH5Zjpegls.so -> libH5Zjpegls.so.2
│           ├── libH5Zjpegls.so.2.0.0
│           └── libH5Zjpegls.so.2 -> libH5Zjpegls.so.2.0.0
├── include
│   ├── fcicomp_jpegls.h
│   └── H5Zjpegls.h
├── lib
│   ├── libfcicomp_jpegls.so -> libfcicomp_jpegls.so.1
│   ├── libfcicomp_jpegls.so.1.0.1
│   └── libfcicomp_jpegls.so.1 -> libfcicomp_jpegls.so.1.0.1
└── share
    └── cmake
        ├── fcicomp-H5Zjpegls_install_manifest.txt
        ├── fcicomp_jpegls
        │   ├── fcicomp_jpegls-config.cmake
        │   ├── fcicomp_jpegls-config-version.cmake
        │   ├── fcicomp_jpegls-targets.cmake
        │   └── fcicomp_jpegls-targets-release.cmake
        ├── fcicomp-jpegls_install_manifest.txt
        └── h5zjpegls
            ├── h5zjpegls-config.cmake
            ├── h5zjpegls-config-version.cmake
            ├── h5zjpegls-targets.cmake
            └── h5zjpegls-targets-release.cmake
```

2.5.4 Post-installation environment configuration

If the FCI decompression software is not deployed at standard location, users will have to make their LD_LIBRARY_PATH and HDF5_PLUGIN_PATH environment variable point to the locations where the FCI decompression software is installed using the following commands. Replace ${FCIDECOMP_INSTALLATION_PATH} by the path where the FCI decompression software is installed on the system.
Users may add the FCI decompression-related paths at the beginning of each path depending on previous versions of programs and libraries already deployed on their systems.
For Bourne, bash, and related shells:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${FCIDECOMP_INSTALLATION_PATH}/lib:

export
          HDF5_PLUGIN_PATH=$HDF5_PLUGIN_PATH:${FCIDECOMP_INSTALLATION_PATH}/hd
          f5/lib/plugin/
```

For csh and related shells:

```
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:${FCIDECOMP_INSTALLATION_PATH}/lib
setenv HDF5_PLUGIN_PATH
          $HDF5_PLUGIN_PATH:${FCIDECOMP_INSTALLATION_PATH}/hdf5/lib/plugin
```

2.5.5 Post-installation test

Make sure that the netCDF-C library is installed on your system and that ncdump utility is in your PATH environment variable. For this run

```
ncdump
```

This should print ncdump utility help. Otherwise, try to modify your PATH and LD_LIBRARY_PATH: for Bourne, bash, and related shells:

```
export PATH=${NETCDF_INSTALLATION_PATH}/bin/:$PATH
export LD_LIBRARY_PATH${NETCDF_INSTALLATION_PATH}/lib:
          ${HDF5_INSTALLATION_PATH}/lib:$LD_LIBRARY_PATH
```

For csh and related shells:

```
setenv PATH ${NETCDF_INSTALLATION_PATH}/bin/:$PATH
setenv LD_LIBRARY_PATH
          ${NETCDF_INSTALLATION_PATH}/lib${HDF5_INSTALLATION_PATH}/lib:$LD_LI
          BRARY_PATH
```

When ncdump can be run, run the postInstallationTest.sh script to check that the installation is successful:

```
./fcidecomp-test/postInstallationTest.sh
```

This script uses ncdump utility to read the compressed data inside a compressed netCDF file and checks that the data are correct by comparing the extracted data with reference data.

2.6 Software upgrade

When installing the FCI decompression software modules, symbolic links are created on the libraries.
To upgrade the FCI decompression software, the same installation procedure as the one described in section 2.5.3 can be followed.
For example, consider that the fcicomp-jpegls module has been deployed in build version 1.0.0 on the system with the installation path being “/usr/local/fcicomp/”. Then, the following files have been created into the directory“/usr/local/fcicomp/lib/”:

```
libfcicomp_jpegls.so -> libfcicomp_jpegls.so.1
libfcicomp_jpegls.so.1 -> libfcicomp_jpegls.so.1.0.0
libfcicomp_jpegls.so.1.0.0
```

The “->” represent the symbolic links. The file “libfcicomp_jpegls.so” is a symbolic link pointing on the library “so name” symbolic link “libfcicomp_jpegls.so.1” (API version) and in turn on the library “real name” file “libfcicomp_jpegls.so.1.0.0” (build version).
Now, consider that the fcicomp-jpegls module needs to be upgraded in version 1.1.0. The same installation procedure as the one described in section 2.5.3.1 can be followed with the installation path being “/usr/local/fcicomp/”. The directory “/usr/local/fcicomp/lib/” will now contain the following files.

```
libfcicomp_jpegls.so -> libfcicomp_jpegls.so.1
libfcicomp_jpegls.so.1 -> libfcicomp_jpegls.so.1.1.0
libfcicomp_jpegls.so.1.0.0
libfcicomp_jpegls.so.1.1.0
```

The library “so name” symbolic link “libfcicomp_jpegls.so.1” now points on the new version of the "fcicomp-jpegls library “real name” file: “libfcicomp_jpegls.so.1.1.0”.
Note that the previous version of the library has not been removed. It could be restored by making the library “so name” symbolic link “libfcicomp_jpegls.so.1” pointing on the previous build version “libfcicomp_jpegls.so.1.1.0”.
The same applies for the other dynamic libraries of the FCI decompression software:

- fcicomp-jpegls
- fcicomp-H5Zjpegls

2.7 Removal of the software

The software removal procedure consists in removing the files that have been installed on the system.
For each module, the files that have been deployed on the system are listed into the install manifest file in the ${FCIDECOMP_INSTALLATION_PATH}/share/cmake/” directory.
To uninstall the FCI decompression software, use the following commands. Use “sudo” before those commands to uninstall modules that have been installed in directories that require sudo privileges.

```
xargs rm < ${FCIDECOMP_INSTALLATION_PATH}/share/cmake/fcicomp-
         H5Zjpegls_install_manifest.txt
```

```
xargs rm < ${FCIDECOMP_INSTALLATION_PATH}/share/cmake/fcicomp-
         jpegls_install_manifest.txt
```

This removes the files that have been installed, including the symbolic links but not the directories.
Users can remove the full installation directory with their own responsibility taking care of any other software that might have been installed in that directory. In the following command line, replace ${FCIDECOMP_INSTALLATION_PATH} path by the path where the software has been installed. Use “sudo” before rm if the installation path is a directory that requires sudo privileges.

```
rm -rf ${FCIDECOMP_INSTALLATION_PATH}
```

The environment variables PATH, LD_LIBRARY_PATH, and HDF5_PLUGIN_PATH also need to be restored if they were modified during the post-installation environment configuration (section 2.5.4).

ANNEX A COTS BUILDING PROCEDURE

zlib 1.2.8

Zlib is a lossless data-compression library that implements the "deflate" method. This library is required by HDF5. Below is described a procedure to build zlib.
Download zlib source code, version 1.2.8, tar.gz from here: http://www.zlib.net/
Extract the source code:

```
tar zxvf zlib-1.2.8.tar.gz
cd zlib-1.2.8
```

Define the installation target directory. For example, in the command line below the <zlib_install_directory> could be replaced by “/usr/local/FCICOMP_COTS/zlib/V_1_2_8/”.

```
INSTALL=<zlib_install_directory>
```

Build and install zlib:

```
./configure --prefix=$INSTALL
make install
```

zlib is installed in the <zlib_install_directory>.

HDF5 1.8.14

HDF5 is Data model, library, and file format for storing and managing data. It is the storage layer used by netCDF-4. Below is described a procedure to build HDF5.

Install zlib as described in the previous section.
Download HDF5 source code, version 1.8.15, tar.gz from here:
http://www.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8.14/src/hdf5-1.8.14.tar.gz
Extract the source code:

```
tar zxvf hdf5-1.8.14.tar.gz
cd hdf5-1.8.14
```

Define the installation target directory. For example, in the command line below the <hdf5_install_directory> could be replaced by “/usr/local/FCICOMP_COTS/hdf5/V_1_8_14/”.

```
INSTALL=<hdf5_install_directory>
```

Define the location where zlib can be found:

```
ZLIB=<zlib_install_directory>
```

Enable the dynamic linking library.

```
export LIBS=-ldl
```

Build and install HDF5 with the high level library enable:

```
./configure --enable-hl --with-zlib=$ZLIB --prefix=$INSTALL
make
make install
```

HDF5 is installed in the <hdf5_install_directory>.

netCDF 4.3.2

netCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. Below is described a procedure to build netCDF.

Install zlib and HDF5 as described in the previous sections.
Download netCDF C source code, version 4.3.2, tar.gz from here:
https://github.com/Unidata/netcdf-c/archive/v4.3.2.tar.gz
Extract the source code:

```
tar zxvf netcdf-c-4.3.2.tar.gz
cd netcdf-c-4.3.2
```

Define the installation target directory. For example, in the command line below the <netcdf_install_directory> could be replaced by "/usr/local/FCICOMP_COTS/netcdf/V_4_3_2/".

```
INSTALL=<netcdf_install_directory>
```

Define the location where zlib and HDF5 can be found:

```
ZLIB=<zlib_install_directory>
HDF5=<hdf5_install_directory>
```

Setup the LD_LIBRARY_PATH, the LDFLAGS and CPPFLAGS environment variables

```
export LD_LIBRARY_PATH=$HDF5/lib:$ZLIB/lib:$LD_LIBRARY_PATH
export LDFLAGS="-L$HDF5/lib -L$ZLIB/lib"
export CPPFLAGS="-I$HDF5/include -I$ZLIB/include"
```

Build and install netCDF with the netCDF-4 library enable:

```
./configure --enable-netcdf-4 --prefix=$INSTALL
make
make install
```

netCDF is installed in the <netcdf_install_directory>.

CharLS 1.0

CharLS is an optimized implementation of the JPEG-LS standard for lossless and near-lossless image compression. Below is described a procedure to build CharLS.

CMake must be installed on the system before building CharLS.
Download CharLS source code, version 1.0, zip from here: http://charls.codeplex.com/
Extract the source code:

```
unzip CharLS-source-1.0.zip -d CharLS-source-1.0
cd CharLS-source-1.0
```

Patch CharLS source code before compiling (see https://charls.codeplex.com/workitem/7823):

```
sed -i.ori '1s/^/#include "header.h"\n/' defaulttraits.h
```

This add the line " #include "header.h" " into the file "defaulttraits.h".
If sed is not available on the system, this operation can be manually performed editing the file "defaulttraits.h" with a text editor and adding the following line at the beginning of this file.

```
#include "header.h
```

Create a building directory "CharLS-build-1.0", move into it and build CharLS shared library in release mode:

```
mkdir ../CharLS-build-1.0
cd ../CharLS-build-1.0
cmake -Dcharls_BUILD_SHARED_LIBS=ON -DCMAKE_BUILD_TYPE=RELEASE \
       ../CharLS-source-1.0
make
```

Define the installation target directory. For example, in the command line below the <charls_install_directory> could be replaced by "/usr/local/FCICOMP_COTS/charls/V_1_0/".

```
INSTALL=<charls_install_directory>
```

Create the install directory tree and copy the required files into that directory:

```
mkdir -p $INSTALL/lib $INSTALL/include
cp libCharLS.so $INSTALL/lib
cd ../CharLS-source-1.0
cp config.h interface.h publictypes.h $INSTALL/include
```

CharLS library is now installed in the <charls_install_directory>.

END OF DOCUMENT