EARTH OBSERVATION MISSION CFI SOFTWARE

Release Notes –Version 4.6

1 INTRODUCTION

This document describes the changes introduced in this release of the Earth Observation Mission CFI Software.

2 RELEASE DESCRIPTION

2.1 Software

The following table lists the released libraries, their version and issue date:

Library Name	Version	Issue Date
File Handling	4.6	3 October 2013
Data Handling	4.6	3 October 2013
Lib	4.6	3 October 2013
Orbit	4.6	3 October 2013
Pointing	4.6	3 October 2013
Visibility	4.6	3 October 2013
EECommon (*)	4.6	3 October 2013

(*) only C++ and JAVA APIs

The core API of the above libraries is written in C and provides an API for C, C++ and JAVA.

The libraries installation packages are available for download at the following URL (registration required):

http://eop-cfi.esa.int/index.php/mission-cfi-software/eocfi-software/branch-4-x/eocfi-v4x-download

2.2 Documentation

The following documents are available:

Type	Document Name	Version
General	Mission Conventions Document	4.6
General	General Software User Manual	4.6
C API	Quick Start Guide	4.6
C API	File Handling Software User Manual	4.6
C API	Data Handling Software User Manual	4.6
C API	Lib Software User Manual	4.6
C API	Orbit Software User Manual	4.6
C API	Pointing Software User Manual	4.6
C API	Visibility Software User Manual	4.6

The documentation is available for download and on-line browsing at the following URL:
http://eop-cfi.esa.int/index.php/mission-cfi-software/eocfi-software/branch-4-x/eocfi-v4x-documentation

More information on the Earth Observation CFI Software can be found at the following URL:
http://eop-cfi.esa.int/index.php/mission-cfi-software/eocfi-software

2.3 Supported platforms

The following platforms are supported by this release of the CFI
(the following are requirements for the **C API**):

- **LINUX32_LEGACY**
  - LINUX 32-bits (Legacy)
  - Platform Requirements: x86 based PC, Linux Operating System (Kernel version 2.6.x)
  - Software Requirements: gcc compiler version 4.2.x, glibc (C Library) version 2.7

- **LINUX64_LEGACY**
  - LINUX 64-bits (Legacy)
  - Platform Requirements: x86_64 based PC, Linux Operating System (Kernel version 2.6.x)

- Software Requirements: gcc compiler version 4.2.x, glibc (C Library) version 2.7

- **LINUX64**
  - LINUX 64-bits
  - Platform Requirements: x86_64 based PC, Linux Operating System (Kernel version 2.6.x)
  - Software Requirements: gcc compiler version 4.5.x, glibc (C Library) version 2.12

- **WINDOWS**
  - Microsoft WINDOWS PC (32-bits)
  - Platform Requirements: x86 based PC, Microsoft Windows XP Operating Systems.
  - Software Requirements: Microsoft Visual C++ Compiler (Visual Studio 2008 Professional)

- **MACIN64**
  - MACOSX on Intel (64-bits)
  - Platform Requirements: x86_64 based Mac Computer, Mac OS X version 10.5.x
  - Software Requirements: gcc compiler version 4.2.x

The following are additional requirements for the **C++ API** (a C++ compiler is required):

- g++ compiler version 4.2.x for LINUX32_LEGACY, LINUX64_LEGACY, MACIN64
- g++ compiler version 4.5.x for LINUX64
- Microsoft Visual C++ Compiler (Visual Studio 2008) for WINDOWS

The following are additional requirements for the **JAVA API** (a JAVA SDK is required):

- Java Standard Edition (SE) version 6 for all platforms

2.4 Installation Packages

The CFI libraries are provided as zip packages:

API	Package Name	MD5 Checksum
C	EOCFI-4.6-CLIB-LINUX32_LEGACY.zip	e727745cc5054841a53b82a081683105
C	EOCFI-4.6-CLIB-LINUX64.zip	e2da56b2bef43594a82dd12426da667c
C	EOCFI-4.6-CLIB-LINUX64_LEGACY.zip	df9d723de0b7406a9d4b2cdae5510fab

C	EOCFI-4.6-CLIB-MACIN64.zip	a0d9bbe554be396a40459107acb9eb63
C	EOCFI-4.6-CLIB-WINDOWS.zip	f1363df0a3e7ceaa55322c64c8bf79ec
C++	EOCFI-4.6-CPPLIB-LINUX32_LEGACY.zip	df5b0e6f753305cfd7f1c894bbce9017
C++	EOCFI-4.6-CPPLIB-LINUX64.zip	77b53abc092600e98adae10e4cf4d8bb
C++	EOCFI-4.6-CPPLIB-LINUX64_LEGACY.zip	bed451767285dcc64c4498a5b44bb707
C++	EOCFI-4.6-CPPLIB-MACIN64.zip	a604ad78f062ab01d219185ab893cf04
C++	EOCFI-4.6-CPPLIB-WINDOWS_DLL.zip	9e8f1240bf515050adb8f2be70dc3583
C++	EOCFI-4.6-CPPLIB-WINDOWS_STA.zip	60d4d2e3dcb0cd72a17e5dd767600d5d
JAVA	EOCFI-4.6-JAVALIB-LINUX32_LEGACY.zip	f64d7b1a9a780a8ab53379b2f2d61ae1
JAVA	EOCFI-4.6-JAVALIB-LINUX64.zip	67eb75960f65b4a4f55b66ec1f8d14a6
JAVA	EOCFI-4.6-JAVALIB-LINUX64_LEGACY.zip	a21c5119c36a13371e01d8ef4ca2785c
JAVA	EOCFI-4.6-JAVALIB-MACIN64.zip	884be4dec8dcb8bebdc71b139f06553b
JAVA	EOCFI-4.6-JAVALIB-WINDOWS.zip	ae1406dae08bedbd787293454e384014

(*) Dynamic libraries (DLLs)

(**) Static libraries

DEM datasets are distributed separately and are available for download at the following URL:

http://eop-cfi.esa.int/index.php/mission-cfi-software/eocfi-software/support-files

2.5 Installation Hints

The CFI libraries can be installed by expanding the installation package in any directory.

For specific hints related to the usage of the libraries, please consult Section 6 “CFI LIBRARIES INSTALLATION” of the General SUM and Section 6 “LIBRARY USAGE” of each Library User Manual.

In order to be able to use the XML validation function in the explorer_data_handling library, it is necessary to install the xerces libraries and the SAX2Count binary. The PATH environment variable shall be pointing at the SAX2Count location.

As of version 4.3, dynamic linking to libxml2 external libraries is no longer required.

As of version 4.5, user applications using the pointing library need to be built with openmp support (adding –fopenmp switch in gcc).

3 NEW FEATURES

The following sections describe the new features introduced in this release.

The description refers to the C API. Equivalent features and methods are available in the C++ and JAVA APIs.

3.1 Data Handling Library

- **New Attitude configuration file**: it is now possible to define satellite attitudes (nominal, satellite, instrument) via an attitude definition file. The attitude definition file can be read and loaded in a data structure of type **xd_attitude_definition_data** via the new function **xd_read_att_def**. The new function **xd_write_att_def** can be used instead to write such data structure to file. Note that such data structure can be used to initialize the correspondent attitude ids (see section 3.3). The benefit of this feature is that it is possible to define the satellite attitude via a configuration file instead of hard coding it in the user application and a modification in the attitude definition does not require changing the user application.
  Equivalent methods are available in the C++ and Java API: see **AttitudeDefinitionModel** Class.

3.2 Orbit Library

- **New orbit_id initialization methods:**
  - XO_ORBIT_INIT_USER_OSV_LIST_MODE: the orbit_id is initialized with a list of generic state vectors.
  - XO_ORBIT_INIT_TLE_SDP4_MODE: SDP4 TLE propagator (see below).

- **SDP4 TLE propagator supported:** this propagator is suited for satellite having orbital period longer than 225 minutes.

- **Fit mode in TLE generation:** (**xo_osv_to_tle** and **xo_gen_tle** functions) it is now possible to generate a TLE record based on a generic list of State Vectors, a Predicted Orbit File (POF) or a Restituted Orbit File (ROF). The TLE is generated in such a way that the trajectory computed using the TLE propagator is as close as possible to the input list of OSVs or to the trajectory computing by propagating the POF or interpolating the ROF.

- Executable tool to generate TLE files: the **gen_tle** tool provides the same features as xo_gen_tle function plus the possibility to generate a TLE file with multiple entries.

3.3 Pointing Library

- **Runtime performance improvements:** a few changes have been made to improve the runtime performance on geo-location functions using DEMs. This is beneficial for user applications processing a large number of data for which runtime is crucial (e.g. the Sentinel-3 IPF).

- **New method to initialize attitude ids:** it is now possible to initialize the three attitude ids (nominal, satellite, instrument) via a new function **xp_attitude_define** that receives as input a data structure of type **xd_attitude_definition_data** (see section 3.1).
  The equivalent method is available in the C++ and Java API: see **attitudeDefine** in the **PointingFunc** class.

3.4 Visibility Library

- **Swath_id**: it is now possible to handle swath data by initializing a swath_id object that can be later manipulated and passed to visibility functions. The swath_id can be initialized via the new function

xv_swath_id_init. Accessory functions are: **xv_swath_id_close, xv_swath_get_id_data; xv_swath_set_id_data**.
Equivalent methods are available in the C++ and Java API: see **swathId** class.

- **New visibility functions**: New visibility functions **xv_zonevistime_compute** and **xv_stationvistime_compute**. The main improvement w.r.t. already existing functions is that they receive as input a swath_id for the swath definition and attitude ids for the attitude definition. Accessory functions are: **xv_timesegments_compute_not/or/end/sort/merge/delta/mapping** for segment manipulation; **xv_swathpos_compute** for swath computation.
Equivalent methods are available in the C++ and Java API: see **Swath** class.

4 SOLVED PROBLEMS

The following Anomalies have been solved:

ANR Id	Description
420	Discontinuities using xp_target_altitude.
428	Outliers in DEM intersection when atmosphere refraction is used.
521	Visibility Lib, function xv_zone_vis_time: incorrect computation when zone segment is on the equator. *This issue had been reported by CryoSat-2 PDGS.*

5 KNOWN PROBLEMS

The following anomalies will be fixed in a future release:

ANR Id	Description
554	xo_gen_tle: validity encoded in filename is not compliant with file format standard.

6 USER SUPPORT

For any question related to the usage of the EOCFI or to report a problem, please contact:

EOCFI Software Support Team

email: eocfi@eopp.esa.int