Earth Explorer
Mission CFI Software

EXPLORER_DATA_HANDLING
SOFTWARE USER MANUAL

Code:	CS-MA-DMS-GS-0009
Issue:	3.5
Date:	26/05/06

	Name	Function	Signature
Prepared by:	Juan José Borrego Bote	Project Engineer	
Checked by:	José Antonio González Abeytua	Project Manager	
Approved by:	José Antonio González Abeytua	Project Manager	

DEIMOS Space S.L.
Ronda de Poniente, 19
Edificio Fiteni VI, Portal 2, 2ª Planta
28760 Tres Cantos (Madrid), SPAIN
Tel.: +34 91 806 34 50
Fax: +34 91 806 34 51
E-mail: deimos@deimos-space.com

Document Information

Contract Data		Classification	
Contract Number:	15583/01/NL/GS	Internal	
		Public	
Contract Issuer:	ESA / ESTEC	Industry	X
		Confidential	

External Distribution		
Name	Organisation	Copies

Electronic handling	
Word Processor:	Adobe Framemaker 6.0
Archive Code:	P/SUM/DMS/01/026-044
Electronic file name:	expcfi-dms-sum-007-10

Document Status Log

Issue	Change Description	Date	Approval
3.4	New document. Library first version. Issue number corresponds to CFI library issue	18/11/05	
3.5	• Maintenance release. • New features: - function xd_xml_validate	26/05/06	

Table of Contents

List of Tables

1 SCOPE

The EXPLORER_DATA_HANDLING Software User Manual provides a detailed description of usage of the CFI functions included within the EXPLORER_DATA_HANDLING CFI software library.

2 ACRONYMS AND NOMENCLATURE

2.1 Acronyms

ANX	Ascending Node Crossing
AOCS	Attitude and Orbit Control Subsystem
ASCII	American Standard Code for Information Interchange
BOM	Beginning Of Mission
CFI	Customer Furnished Item
EOM	End Of Mission
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
GPL	GNU Public License
GPS	Global Positioning System
IERS	International Earth Rotation Service
I/F	Interface
LS	Leap Second
OBT	On-board Binary Time
OSF	Orbit Scenario File
SRAR	Satellite Relative Actual Reference
SUM	Software User Manual
TAI	International Atomic Time
UTC	Coordinated Universal Time
UT1	Universal Time UT1
WGS[84]	World Geodetic System 1984

2.2 Nomenclature

CFI	A group of CFI functions, and related software and documentation. that will be distributed by ESA to the users as an independent unit
CFI function	A single function within a CFI that can be called by the user
Library	A software library containing all the CFI functions included within a CFI plus the supporting functions used by those CFI functions (transparently to the user)

3 APPLICABLE AND REFERENCE DOCUMENTS

3.1 Applicable Documents

[GEN_SUM]	Earth Explorer Mission CFI Software. General Software User Manual. CS-MA-DMS-GS-0002. Issue 3.5. 26/05/06
[EE_FMT]	Earth Explorer File Format Standards. PE-TN-ESA-GS-0001 Issue 1.4 13/06/04
[GS_FMT]	Cryosat Ground Segment Mission Files Format Specification. CS-ID-ESA-GS-0224
[PDS_FMT]	Cryosat Ground Segment Payload Data Segment L0 Product Specitication Format CS-ID-ACS-GS-0119

3.2 Reference Documents

[MCD]	Earth Explorer Mission CFI Software. Mission Conventions Document. CS-MA-DMS-GS-0001. Issue 1.3. 15/07/03.
[F_H_SUM]	Earth Explorer Mission CFI Software. EXPLORER_FILE_HANDLING Software User Manual. CS-MA-DMS-GS-0008. Issue 3.5. 26/05/06.
[LIB_SUM]	Earth Explorer Mission CFI Software. EXPLORER_LIB Software User Manual. CS-MA-DMS-GS-0003. Issue 3.5. 26/05/06.
[ORBIT_SUM]	Earth Explorer Mission CFI Software. EXPLORER_ORBIT Software User Manual. CS-MA-DMS-GS-0004. Issue 3.5. 26/05/06.
[POINT_SUM]	Earth Explorer Mission CFI Software. EXPLORER_POINTING Software User Manual. CS-MA-DMS-GS-0005. Issue 3.5. 26/05/06.
[VISIB_SUM]	Earth Explorer Mission CFI Software. EXPLORER_VISIBILITY Software User Manual. CS-MA-DMS-GS-0006. Issue 3.5. 26/05/06.
[IERS]	http://www.iers.org/iers/publications/bulletins/

4 INTRODUCTION

4.1 Functions Overview

This software library contains a set of functions for reading and writing Earth Explorer Mission Files.

The following CFI functions are included:

4.1.1 Reading routines

- **xd_read_fhr:** reads the fixed header for an Earth Explorer XML file.
- **xd_read_bulletin:** reads the time correlations from an IERS bulletin.
- **xd_read_orbit_file**: reads orbit files consisting in a list of state vectors of the satellite in the orbit.The following files are supported: Predicted Orbit files, Restituted Orbit files and DORIS Preliminary files.
- **xd_read_doris**: reads DORIS Navigator files for CRYOSAT.
- **xd_read_doris_header:** reads the MPH and SPH data from a DORIS Navigator file for CRYOSAT.
- **xd_read_osf**: reads Orbit Scenario files.
- **xd_read_att:** reads a generic attitude file.
- **xd_read_star_tracker:** reads an star traker file for CRYOSAT.
- **xd_read_star_tracker_conf_file:** reads an star tracker configuration file for CRYOSAT.
- **xd_read_dem:** provides the points of a DEM that are adjacent to a given point.
- **xd_read_dem_config_file:** reads a DEM configuration file.
- **xd_read_sdf**: reads swath definition files.
- **xd_read_stf:** reads swath template files.
- **xd_read_stf_vhr:** reads the variable header for swath template files
- **xd_read_zone:** reads the parameters of one zone in a zone database file.
- **xd_read_zone_file:** reads a zone database file.
- **xd_read_zone_id:** reads the list of zone names from a zone database file.
- **xd_read_station:** reads the parameters of one station in a station database file.
- **xd_read_station_file:** reads a station database file.
- **xd_read_station_id:** reads the list of station names from a station database file
- **xd_read_star:** reads the parameters of one star in a star database file.
- **xd_read_star_file:** reads a star database file.
- **xd_read_star_id:** reads the list of star id. from a star database file

4.1.2 Writing routines

- **xd_write_orbit_file**: writes an orbit file using as input an structure with the data of the file
- **xd_write_osf**: writes an orbit scenario file using as input an structure with the data of the file
- **xd_write_doris:** writes a DORIS Navigator file.
- **xd_write_att:** writes a generic attitude file.

- **xd_write_stf**: writes a swath template file using as input the data structure containing the data for the swath.

4.1.3 Validation of XML files

- **xd_xml_validate**: validates an XML file using an XML schema as reference.

5 LIBRARY INSTALLATION

For a detailed description of the installation of any CFI library, please refer to [GEN_SUM].

6 LIBRARY USAGE

Note that to use the EXPLORER_DATA_HANDLING software library, the following other CFI software libraries are required:

- `EXPLORER_FILE_HANDLING` (See [F_H_SUM]).

It is also needed to have properly installed in the system the following external GPL libary:

- `LIBXML2` (see [GEN_SUM]).

and the POSIX thread library:

- libpthread.so (pthread.lib for WINDOWS)

To use the EXPLORER_DATA_HANDLING software library in a user application, that application must include in its source code either:

- `explorer_data_handling.h` (for a C application)
- `explorer_data_handling.inc` (for a ForTran application under SOLARIS)
- `explorer_data_handling_win.inc` (for a ForTran application under Windows 95/NT)

To link correctly this application, the user must include in his linking command flags like (assuming *cfi_lib_dir* and *cfi_include_dir* are the directories where respectively all CFI libraries and include files have been installed, see [GEN_SUM] for installation procedures):

- SOLARIS/LINUX:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_data_handling
                  -lexplorer_file_handling -lxml2 -lpthread
```

- WINDOWS:

```
/I "cfi_include_dir"  /libpath:"cfi_lib_dir"
                                libexplorer_data_handling.lib
                                libexplorer_file_handling.lib
                                libxml2.lib pthread.lib
```

- MacOS:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_data_handling
                  -lexplorer_file_handling -lpthread
                  -framework libxml -framework libiconv
```

All functions described in this document have a name starting with the prefix `xd_`

To avoid problems in linking a user application with the `EXPLORER_DATA_HANDLING` software library due to the existence of names multiple defined, the user application should avoid naming any global software item beginning with either the prefix `XD_` or `xd_`.

It is possible to call the following CFI functions from a user application.

Table 1: CFI functions included within EXPLORER_DATA_HANDLING library

Function Name	Enumeration value	Long
Main CFI Functions		
xd_read_fhr	XD_READ_FHR_ID	0

Function Name	Enumeration value	Long
xd_read_bulletin	XD_READ_BULLETIN_ID	1
xd_read_orbit_file	XD_READ_ORBIT_FILE_ID	2
xd_read_doris	XD_READ_DORIS_ID	3
xd_read_doris_header	XD_READ_DORIS_HEADER_ID	4
xd_read_osf	XD_READ_OSF_ID	5
xd_read_sdf	XD_READ_SDF_ID	6
xd_read_stf	XD_READ_STF_ID	7
xd_read_stf_vhr	XD_READ_STF_VHR_ID	8
xd_read_att	XD_READ_ATT	9
xd_read_star_tracker	XD_READ_STAR_TRACKER_ID	10
xd_read_str_conf_file	XD_READ_STR_CONF_FILE_ID	11
xd_read_dem_config_file	XD_READ_DEM_CONFIG_FILE_ID	12
xd_read_dem	XD_READ_DEM_ID	13
xd_read_star	XD_READ_STAR_ID	14
xd_read_star_file	XD_READ_STAR_FILE_ID	15
xd_read_star_id	XD_READ_STAR_ID_ID	16
xd_read_station	XD_READ_STATION_ID	17
xd_read_station_file	XD_READ_STATION_FILE_ID	18
xd_read_station_id	XD_READ_STATION_ID_ID	19
xd_read_zone	XD_READ_ZONE_ID	20
xd_read_zone_file	XD_READ_ZONE_FILE_ID	21
xd_read_zone_id	XD_READ_ZONE_ID_ID	22
xd_write_orbit_file	XD_WRITE_ORBIT_FILE_ID	23
xd_write_doris	XD_WRITE_DORIS_ID	24
xd_write_osf	XD_WRITE_OSF_ID	25
xd_write_stf	XD_WRITE_STF_ID	26
xd_write_att	XD_WRITE_ATT_ID	27
xd_xml_validate	XD_XML_VALIDATE_ID	28
Error Handling Functions		

Function Name	Enumeration value	Long
xd_verbose	not applicable	
xd_silent		
xd_get_code		
xd_get_msg		
xd_print_msg		

Notes about the table:

- To transform the extended status flag returned by a CFI function to either a list of error codes or a list of error messages, the enumeration value (or the corresponding long value) described in the table must be used
- The error handling functions have no enumerated values

Whenever available **it is strongly recommended to use enumeration values rather than integer values**.

6.1 Usage hints

Every CFI function has a different length of the Error Vector, used in the calling I/F examples of this SUM and defined at the beginning of the library header file. In order to provide the user with a single value that could be used as Error Vector length for every function, a generic value has been defined (XD_ERR_VECTOR_MAX_LENGTH) as the maximum of all the Error Vector lengths. This value can therefore be safely used for every call of functions of this library.

6.2 General Enumerations

The aim of the current section is to present the enumeration values that can be used rather than integer parameters for some of the input parameters of the EXPLORER_DATA_HANDLING routines, as shown in the table below. The enumerations presented in [GEN_SUM] are also applicable.

Table 2: Enumerations within EXPLORER_DATA_HANDLING library

Input	Description	Enumeration value	Long
Boolean values	False value	XD_FALSE	0
	True value	XD_TRUE	1
Returned status code	Error	XD_ERR	-1
	Ok status	XD_OK	0
	Warnig	XD_WARN	1

Table 2: Enumerations within EXPLORER_DATA_HANDLING library

Input	Description	Enumeration value	Long
Satellite ID	Default Satellite 0	XD_SAT_DEFAULT	0
	Default Satellite 1	XD_SAT_DEFAULT1	1
	Default Satellite 2	XD_SAT_DEFAULT2	2
	Default Satellite 3	XD_SAT_DEFAULT3	3
	Default Satellite 4	XD_SAT_DEFAULT4	4
	Default Satellite 5	XD_SAT_DEFAULT5	5
	Default Satellite 6	XD_SAT_DEFAULT6	6
	Default Satellite 7	XD_SAT_DEFAULT7	7
	Default Satellite 8	XD_SAT_DEFAULT8	8
	Default Satellite 9	XD_SAT_DEFAULT9	9
	ERS 1	XD_SAT_ERS1	11
	ERS 2	XD_SAT_ERS2	12
	EnviSat	XD_SAT_ENVISAT	21
	Metop 1	XD_SAT_METOP1	31
	Metop 2	XD_SAT_METOP2	32
	Metop 3	XD_SAT_METOP3	33
	CryoSat	XD_SAT_CRYOSAT	41
	ADM	XD_SAT_ADM	51
	GOCE	XD_SAT_GOCE	61
	SMOS	XD_SAT_SMOS	71
	Terrasar	XD_SAT_TERRASAR	81
Time initialization mode	Select the whole file	XD_SEL_FILE	0
	Select a time range	XD_SEL_TIME	1
	Select an orbit range	XD_SEL_ORBIT	2
	Select the default value	XD_SEL_DEFAULT	3
Time reference	Undefined	XD_TIME_UNDEF	-1
	TAI	XD_TIME_TAI	0
	UTC	XD_TIME_UTC	1
	UT1	XD_TIME_UT1	2
	GPS	XD_TIME_GPS	3
Attitude data type	Quaternions	XD_ATT_QUATERNIONS	0
	Angles	XD_ATT_ANGLES	1

Table 2: Enumerations within EXPLORER_DATA_HANDLING library

Input	Description	Enumeration value	Long
Ray tracing model		XD_NO_REF	0
		XD_STD_REF	1
		XD_USER_REF	2
		XD_PRED_REF	3
		XD_STD_REF_N	10
		XD_USER_REF_N	20
		XD_PRED_REF_N	30
		XD_US76_REF	300
		XD_TROPIC_REF	301
		XD_MID_SUM_REF	302
		XD_MID_WIN_REF	303
		XD_SUBAR_SUM_REF	304
		XD_SUBAR_WIN_REF	305
		XD_LUT_REF	400
		XD_US76_REF_N	3000
		XD_TROPIC_REF_N	3001
		XD_MID_SUM_REF_N	3002
		XD_MID_WIN_REF_N	3003
		XD_SUBAR_SUM_REF_N	3004
		XD_SUBAR_WIN_REF_N	3005
		XD_LUT_REF_N	4000
Swath Types		XD_UNKNOWN_SWATH	-1
		XD_POINT_SWATH	0
		XD_LINE_SWATH	1
		XD_INERTIAL_SWATH	2
Asar swath types		XD_NO_ASAR	0
		XD_NARROW_ASAR	1
		XD_WIDE_ASAR	2

Table 2: Enumerations within EXPLORER_DATA_HANDLING library

Input	Description	Enumeration value	Long
Orbit file types	Orbit Scenario File	XD_REF_FILETYPE_OSF	0
	Orbit Event file used as an OSF	XD_REF_FILETYPE_OEF_OSF	1
	FOS Predicted Orbit File	XD_REF_FILETYPE_POF	2
	Orbit Event file used as a POF	XD_REF_FILETYPE_OEF_POF	3
	DORIS Navigator File	XD_REF_FILETYPE_DORIS_NAV	4
	FOS Restituted Orbit File	XD_REF_FILETYPE_ROF	5
	DORIS Preliminary Orbit File	XD_REF_FILETYPE_DORIS_PREM	6
	DORIS Precise Orbit File	XD_REF_FILETYPE_DORIS_PREC	7
Coordinate systems	Barycentric Mean of 2000.0	XD_BAR_MEAN_2000	0
	Heliocentric Mean of 2000.0	XD_HEL_MEAN_2000	1
	Geocentric Mean of 2000.0	XD_GEO_MEAN_2000	2
	Mean of date	XD_MEAN_DATE	3
	True of date	XD_TRUE_DATE	4
	Earth Fixed	XD_EARTH_FIXED	5
	Satellite relative actual reference	XD_SAT_ACT_REF	6
	Quasi-Mean of Date	XD_QUASI_MEAN_DATE	7
	Pseudo-True of Date	XD_PSE_TRUE_DATE	8
	Quasi-True of Date	XD_QUASI_TRUE_DATE	9
	Topocentric	XD_TOPOCENTRIC	10
	Satellite reference	XD_SAT_REF	11
	Satellite relative reference	XD_SAT_REL_REF	12
Attitude reference frames	Orbital reference frame	XD_SAT_ORBITAL_REF	0
	Satellite nominal attitude frame	XD_SAT_NOMINAL_ATT	1
	Satellite attitude frame	XD_SAT_ATT	2
	Instrument attitude frame	XD_INSTR_ATT	3
Different models for DEM	ACE Model	XD_DEM_ACE_MODEL	0
Zone types	zone is not defined as an input and must be read from a file	XD_NOT_DEFINED	-1
	Point zone	XD_POINT	0
	Circular zone	XD_CIRCLE	1
	Segment zone	XD_SEGMENT	2
	Polygonal zone	XD_POLYGON	3

Table 2: Enumerations within EXPLORER_DATA_HANDLING library

Input	Description	Enumeration value	Long
Projection types	Read projection from DB file	XD_READ_DB	0
	Use gnomonic projection	XD_GNOMONIC	1
	Use rectangular projection	XD_RECTANGULAR	2
Validation Status	Invalid file	XD_XML_INVALID	-1
	Valid file	XD_XML_VALID	0

The use of the previous enumeration values could be restricted by the particular usage within the different CFI functions. The actual range to be used is indicated within a dedicated reference named ***allowed range***. When there are not restrictions to be mentioned, the allowed range column is populated with the label ***complete***.

6.3 Data Structures

The aim of this section is to present the data structures that are used in the EXPLORER_DATA_HANDLING library. These structures are used as output/inputs in the reading/writing routines. The following table show the data structures with their names and the data that contains:

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_fhr	Fixed header data	file_name	char [XD_MAX_STR]	File name
		file_description	char [XD_MAX_STR]	File description
		mission	char [XD_MAX_STR]	Mission name
		file_class	char [XD_MAX_STR]	File class
		file_type	char [XD_MAX_STR]	File type
		version	long	File version
		val_start_date	char [32]	Validity start date
		val_stop_date	char [32]	Validity stop date
		system	char [XD_MAX_STR]	System name
		creator	char [XD_MAX_STR]	Creator name
		creator_version	char [XD_MAX_STR]	Creator version
		creation_date	char [32]	Creation date
xd_bulb_table	Data for one entry read from a IERS bulletin	day	double	MJ200 UTC Time
		ut1_utc	double	Difference between UT1 and UTC
		ut1_tai	double	Difference between UT1and TAI
xd_iers_bulletin_b	Data for time correlations read from a IERS bulletin	table1	xd_bulb_table[100]	First table data in the IERS bulletin
		table2	xd_bulb_table[100]	Second table data in the IERS bulletin
xd_time_rec	It contains the time correlations for a given time	tai_time	double	TAI time
		ut1_time	double	UT1 time
		tai_utc	double	Difference between TAI and UTC time
		tai_ut1	double	Difference between TAI and UT1 time
		tai_gps	double	Difference between TAI and GPS time

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_osv_rec	It contains a satellite state vector for a given time	tai_time	double	TAI time for the state vector
		utc_time	double	UTC time for the state vector
		ut1_time	double	UT1 time for the state vector
		abs_orbit	double	Absolute orbit
		pos	dobule[3]	Position vector (x, y, z componets)
		vel	double[3]	Velocity vector (x, y, z componets)
		quality	double	Quality index
xd_orbit_file	Structure for storing the data read from an orbit file	num_rec	long	Number of records
		osv_rec	xd_osv_rec*	Array with the state vectors
xd_doris_file	Structure for storing the data read from a DORIS Navigator file	num_rec	long	Number of records
		osv_rec	xd_osv_rec	State vectors array (EF)
		osv_rec_j2	xd_osv_rec	State vectors array (J2000)
		leap_time	double	Leap time
		leap_sign	int	Leap time sign
		abs_orbit	long	First absolute orbit number
		rel_orbit	long	First relative orbit number

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_doris_mph_sph	Structure for the main and specific product headers	filename	char [XD_MAX_STR]	The description for these fields can be found in [PDS_FMT]
		sensing_start	char [30]	
		sensing_stop	char [30]	
		abs_orbit	long	
		delta_ut1	long	
		rel_orbit	long	
		leap_utc	char [XD_MAX_STR]	
		leap_sign	int	
		leap_err	int	
		num_dsd	long	
		ds_offset	long	
		num_dsr	long	
		proc_stage_code	char [5]	
		ref_doc	char [24]	
		proc_time	char [31]	
		software_version	char [15]	
		phase	char [2]	
		cycle	long	
		state_vector_time	char [31]	
		x_position	double	
		y_position	double	
		z_position	double	
		x_velocity	double	
		y_velocity	double	
		z_velocity	double	
		state_vector_source	char [3]	
		ascii_utc_time_before_leap	double	
		product_err	char [2]	
		tot_size	long	
		num_data_sets	long	

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
		sph_descriptor	char [29]	
		sensing_start_tai	char [31]	
		abs_orbit_start	long	
		rel_time_asc_node_start	double	
		sensing_stop_tai	char [31]	
		abs_orbit_stop	long	
		rel_time_asc_node_stop	double	
		equator_cross_time	char [31]	
		equator_cross_long	long	
		ascending_flag	char [2]	
		start_lat	long	
		start_long	long	
		stop_lat	long	
		stop_long	long	
		num_isps	long	
		num_missing_isps	long	
		num_error_isps	long	
		num_discarded_isps	long	
		num_rs_isps	long	
		num_rs_corrections	long	
		dsr_size	long	

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_osf_rec	It contains the data for an orbital change in an orbit scenario file	abs_orb	long	Absolute orbit number
		rel_orb	long	Relative orbit number
		cycle_days	long	Cycle length in days
		cycle_orbits	long	Number of orbits in a cycle
		mlst	double	Mean local solar time (in hours)
		mlst_drift	double	Mean local solar time drift (seconds per day)
		inclination	double	Orbit inclination
		drift_mode	long	Flag for choosing between inclination of drift model
		anx_tai	double	ANX TAI time
		anx_ut1	double	ANX UT1 time
		anx_utc	double	ANX UTC time
		anx_long	double	ANX longitude
		cycle	long	Cycle number
		phase	long	Phase number
xd_osf_file	Structure for storing the data read from an orbit scenario file	num_rec	long	Number of records
		osf_rec	xd_osf_rec*	Array of state vectors
xd_swath_geometry	It contains the swath geometry	geom_type	long	Geometry type
		az	double[3]	Azimuth points
		el	double[3]	Elevation points
		alt	double[3]	Altitude points
		distance	double[3]	Distance

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_harmonic_data		num_terms	long[3]	Number of harmonics coefficient(pitch, roll and yaw)
		harmonic_type_pitch	long[XD_MAX_NUM_HARMONIC]	Harmonic type
		harmonic_type_roll	long[XD_MAX_NUM_HARMONIC]	Harmonic type
		harmonic_type_yaw	long[XD_MAX_NUM_HARMONIC]	Harmonic type
		harmonic_coef_pitch	double [XD_MAX_NUM_HARMONIC]	Harmonic coefficient
		harmonic_coef_roll	double [XD_MAX_NUM_HARMONIC]	Harmonic coefficient
		harmonic_coef_yaw	double [XD_MAX_NUM_HARMONIC]	Harmonic coefficient
xd_param_model_str		model	long	Model type
		param_num	long	Number of parameters
		model_param	double [XD_NUM_MODEL_PARAM]	Model Parameters
xd_harmonic_model_str		angle_type	long	Angle type
		harmonics	xd_harmonic_data	Harmonic data
		offsets	double [3]	Offsets
xd_file_model_str		num_files	long	Number of files
		files	char **	file list
		aux_file	char *	Auxiliary file
		time_ref	long	Time reference
		time0	double	Start time
		time1	double	Stop time
xd_angle_model_str		angles	double [3]	angles
		offsets	double [3]	offsets
xd_matrix_model_str	Matrix model	att_matrix	double [3][3]	Attitude matrix model
		offsets	double [3]	Offsets
xd_attitutude_model_str	Attitude model structure	attitude_model	long	Attitude model type
		data	Attitude union data	Attitude union. One of the attitude structures.

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
Attitude union data	One of the following attitude structures	AOCS	long	AOCS model
		param_mode	xd_param_model_str	Parameters model
		harmonic_mode	xd_harmonic_model_str	Harmonic model
		file_mode	xd_file_model_str	File model
		angle_mode	xd_angle_model_str	Angle Model
		matrix_mode	xd_matrix_model_str	Matrix Model
xd_asar	ASAR definition structure	asar_type	long	ASAR Swath types
		first_asar_pt	double[2]	Wide/Narrow mode parameters
		second_asar_pt	double[2]	Aditional Wide mode parameters
xd_sdf_rec	Swath Definition data	swath_descr	char [XD_MAX_STR]	Swath description
		swath_id	char [XD_MAX_STR]	Swath_id
		swath_rec_type	long	Algorithm to be used in calculations (XD_Algor_enum)
		num_swath_rec	long	Number of swath records to write in a single OEF
		refr_mode	long	Refraction mode (XD_Target_ray_enum)
		freq	double	Frequency (Hz)
		swath_geom	xd_swath_geometry	Swath geometry
		as	xd_asar	ASAR parameters
		instr_dependency	long	Indicate whether there is instr.dependent data
		sat_nom_att	xd_attitude_model_str *	Attitude data for sat. nominal att
		sat_att	xd_attitude_model_str *	Attitude data for sat. attribute
		instr_att	xd_attitude_model_str *	Attitude data for instrument att
xd_sdf_file	Swath definition file data	num_rec	long	Number of swath records in a SDF
		sdf_rec	xd_sdf_rec *	Swath record data array

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_stf_rec	Swath template record data	left_lat	double	left point latitude
		mid_lat	double	middle point latitude
		right_lat	double	right point latitude
		left_lon	double	left point latitude
		mid_lon	double	middle point latitude
		right_lon	double	right point latitude
xd_stf_vhr	Swath template variable header data	stf_name	char *	swath template file name
		sdf_name	char [XF_MAX_PATH_LENGTH]	Reference swath definition file
		swath_type	XD_Swath_rec_enum	Swath type
		time_step	double	
		refr_mode	long	Refraction model
		freq	double	Frequency (Hz)
		left_alt	double	Left point altitude
		mid_alt	double	middle point altitude
		right_alt	double	right point altitude
		geom_flag	long	true if the geometry of the orbit is defined. False if the state vector at ANX is defined
		rep_cycl	long	repeat cycle
		cycle_length	long	cycle length
		mlst_drift	double	MLST drift
		abs_orbit	long	Absolut orbit
		pos	double [3]	ANX position vector
		vel	double [3]	ANX velocity vector
xd_stf_file	Swath template file data	num_rec	long	number of points in the swath
		vhr	xd_stf_vhr	variable header
		stf_rec	xd_stf_rec *	array with the points in the swath
xd_att_rec	Attitude record	time_ref	long	Time reference
		time	double	time (MJD2000)
		data	double [4]	Quaternions or angles. For angles, the fourth value is dummy

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_att_file	Attitude file data	sat_ref	long	target reference frame
		source_ref	long	initial reference frame: Inertial reference frame
		data_type	long	angles or quaternions (see XD_Attitude_data_type_enum)
		num_rec	long	number of records in the atti-tude lists
		max_gap	double	Maximum time gap between two consequtive records
		att_rec	xd_att_rec*	array with the angle/ quaternion records
xd_tracker_limi ts	star trackers limits data	max_penalty	double	Maximum penalty for the quaternions
		norm_thr	double	Threshold for the modulus of the quaternion
		max_gap	double	Maximum time gap between two consequtive quaternions
xd_tracker_co nf_file	star trackers cofigu-ration file data	aberr_correction	long	Aberration correction flag: -1 = Aberration correction with transposed matrix 0 = No aberration 1 = Aberration correction
		satellite	char [XD_MAX_STR]	Satellite name
		str_limit	xd_tracker_limits	Star tracker limits
		str_att_rot	double [3][3]	Satellite Attitude to star tracker frame rotation matrix
xd_star_tracke r	Star tracker record	quaternion	float[4]	Quaternions
		time	double	MJ2000 in TAI
		status	unsigned char	quaternion status
xd_star_tracke r_file	star tracker file data	str_id	long	Star tracker Id (1,2 or 3)
		num_rec	long	number of lines
		str_rec	xd_star_tracker*	array with the star tracker records

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_dem_ace	DEM configuration data for ACE model	dir	char[100]	Directory where the DEM files are storaged
		res_X	double	Interval between points along X-axis
		res_Y	double	Interval between points along Y-axis
		res_unit	double	Conversion factor from x,y units to the res_X, res_Y units. For example, if res_X is given in seconds and X in degrees then res_unit=3600
		X_num_points	long	Number of points along X-axis (columns)
		Y_num_points	long	Number of points along Y-axis (files)
		x_range	double	longitude of the X-axis for one file (grid).
		x_range	double	longitude of the Y-axis for one file (grid).
		data_size	long	Size in bytes of the data stored in the files
		north_alt	double[4]	Altitude at the North pole cell
		south_alt	double[4]	Altitude at the South pole cell
		offset_x	double	Distance from the middle of a cell to the vertical side.
		offset_y	double	Distance from the middle of a cell to the horizontal side.
xd_dem_config_file	DEM configuration data	model	long	DEM model
		dem_data	xd_dem_ace *	DEM ACE data

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_star_rec	Star data	flag	long	True if the star was found in the star database file.
		star_id	char [XD_MAX_STR]	Star ID
		par	double	Parallax of the star at JD2000 (rads)
		mu_ra	double	RA's proper motion at JD2000 (rad/century)
		mu_dec	double	DEC's proper motion at JD2000 (rad/century)
		rad_vel	double	Radial velocity of the star (au/century)
		star_ra	double	RA of the star at JD2000 (rads)
		star_dec	double	DEC of the star at JD2000 (rads)
xd_star_file	Structure containing all relevant information contained in the star's database file	num_rec	long	Number of stars
		star_rec	xd_star_rec *	Array with all the star data

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_station_rec	Station record data	station_id	char [XD_MAX_STR]	Station ID
		descriptor	char [XD_MAX_STR]	Description of the station
		antenna	char [XD_MAX_STR]	Describes the frequency band in which the anntena works.
		purpose	char [XD_MAX_STR]	Purpose
		type	char [XD_MAX_STR]	Not used.
		num_mask_pt	long	Number of points to define the antenna
		azimuth	double [XD_VERTICES]	Azimuth and elevation defining the antenna mask.
		elevation	double [XD_VERTICES]	
		station_long	double	Station longitude
		station_lat	double	Station latitude
		station_alt	double	Station altitude
		proj_long	double [XD_VERTICES]	longitude/latitude points for the station zone that are equivalent to the set of azimuth/elevation points. The longitude/latitude points are not read from the file but computed in xv_station_vis_time.
		proj_lat	double [XD_VERTICES]	
		points	long	Number of points in the azimuth/elevation and in proj_long/proj_lat arrays.
		long_max	double	Maximum longitude of the station zone
		lat_max	double	Maximum latitude of the station zone
		long_min	double	Minimum longitude of the station zone
		lat_min	double	Minimum latitude of the station zone
xd_station_file		num_rec	long	Number of stations
		station_rec	xd_station_rec *	Array of station records

Table 3: EXPLORER_DATA_HANDLING Structures

Structure name	Description	Structure Data		
		Variable Name	C type	Description
xd_zone_point	Longitude and latitude point	pt_long	double	Longitude
		pt_lat	double	Latitude
xd_zone_rec	Zone record data	zone_id	char [XD_MAX_STR]	Zone ID
		description	char [XD_MAX_STR]	Drescription of the zone
		surface	char [XD_MAX_STR]	Surface type
		creator	char [XD_MAX_STR]	Creator name
		zone_type	XD_Zone_type_enum	Zone type
		projection	long	Projection
		zone_diam	double	Zone diameter in meters. Only used when the ZONE is a POINT zone or a CIRCULAR zone.
		num_points	long	Number of ZONE points (last one, equal to the first one, included)
		zone_point	xd_zone_point *	Array of points of the zone
xd_zone_file	Zone file structure	num_rec	long	Number of zones
		zone_rec	xd_zone_rec *	Array of zone records

7 CFI FUNCTIONS DESCRIPTION

The following sections describe each CFI function.

The calling interfaces are described for both C and ForTran users.

Input and output parameters of each CFI function are described in tables, where C programming language syntax is used to specify:

- Parameter types (e.g. long, double)
- Array sizes of N elements (e.g. param[N])
- Array element M (e.g. [M])

ForTran users should adapt the tables using ForTran syntax equivalent terms:

- Parameter types (e.g. long <=> INTEGER*4, double <=>REAL*8)
- Array sizes of N elements (e.g. param[N] <=> param (N))
- Array element M (e.g. [M] <=> (M+1))

7.1 xd_read_fhr

7.1.1 Overview

The **xd_read_fhr** CFI function reads the fixed header for Earth Explorer XML files.

7.1.2 Calling interface

The calling interface of the **xd_read_fhr** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      char *file_name;
      xd_fhr fhr;
      long ierr[XD_NUM_ERR_READ_FHR];
      status = xd_read_fhr(file_name, &fhr, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      INTEGER*4 IERR(XD_NUM_ERR_READ_FHR), STATUS
      XD_FHR FHR
      STATUS = XD_READ_FHR (FILE_NAME, FHR, IERR)
```

7.1.3 Input parameters

The **xd_read_fhr** CFI function has the following input parameters:

Table 4: Input parameters of xd_read_fhr function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	file name	-	-

7.1.4 Output parameters

The output parameters of the **xd_read_orbit_file** CFI function are:

Table 5: Output parameters of xd_read_fhr function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_fhr	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
Fixed header data	xd_fhr	-	Data structure containing the data read from the fixed header	-	-
ierr	long[]	-	Error vector	-	-

7.1.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_fhr** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_fhr** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 6: Error messages of xd_read_fhr function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open the file	No calculation performed	XD_CFI_READ_FHR_OPEN_FILE_ERR	0
ERR	Error reading the fixed header	No calculation performed	XD_CFI_READ_FHR_GET_FIXED_HEADER_ERR	1

Table 6: Error messages of xd_read_fhr function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error closing the file	No calculation performed	XD_CFI_READ_FHR_CLOSE_FILE_ERR	2

7.1.6 Runtime performances

The following runtime performances have been measured.

Table 7: Runtime performances of xd_read_fhr function

Ultra Sparc II-400 [ms]
TBD

7.2 xd_read_bulletin

7.2.1 Overview

The **xd_read_bulletin** CFI function reads IERS bulletin files and returns the time correlation data.

7.2.2 Calling interface

The calling interface of the **xd_read_bulletin** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      char *bulb_file;
      xd_iers_bulletin_b iers_data
      long ierr[XD_NUM_ERR_READ_BULLETIN];
      status = xd_read_bulletin (bulb_file, &iers_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>
      CHARACTER*LENGTH_NAME BULB_FILE
      INTEGER*4 IERR(XD_NUM_ERR_READ_BULLETIN), STATUS
      XD_IERS_BULLETIN_B IERS_DATA
      STATUS = XD_READ_BULLETIN (BULB_FILE, IERS_DATA, IERR)
```

7.2.3 Input parameters

The **xd_read_bulletin** CFI function has the following input parameters:

Table 8: Input parameters of xd_read_bulletin function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
bulb_file	char*	-	File name	-	-

7.2.4 Output parameters

The output parameters of the **xd_read_bulletin** CFI function are:

Table 9: Output parameters of xd_read_bulletin function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_bulletin	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
IERS bulletin data	xd_iers_bulletin_b	-	Data structure containing the data read from the file	-	-
ierr	long[]	-	Error vector	-	-

7.2.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_bulletin** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_bulletin** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 10: Error messages of xd_read_bulletin function

Error type	Error message	Cause and impact	Error code	Error No
ERR	File does not exist	No calculation performed	XD_CFI_READ_BULLETIN_FILE_ERR	0
ERR	Time table is empty or has wrong format	No calculation performed	XD_CFI_READ_BULLETIN_TABLE_ERR	1

7.2.6 Runtime performances

The following runtime performances have been measured.

Table 11: Runtime performances of xd_read_bulletin function

Ultra Sparc II-400 [ms]
TBD

7.3 xd_read_orbit_file

7.3.1 Overview

The **xd_read_orbit_file** CFI function reads orbit files for Earth Explorer Missions. The files have to be written in XML and consits on a list of state vectors of the satellite along the orbit.

This function can also be used for reading the list of state vectors within Orbit Event files.

7.3.2 Calling interface

The calling interface of the **xd_read_orbit_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      char *file_name;
      long read_fro_flag, time_orbit_flag, time_ref, reading_osv_flag;
      double start_range, stop_range;
      xd_orbit_file orbit_data
      long ierr[XD_NUM_ERR_READ_ORBIT_FILE];
      status = xd_read_orbit_file (file_name, &read_fro_flag,
                                   &time_orbit_flag, &time_ref,
                                   &start_range, &stop_range,
                                   &reading_osv_flag,
                                   &orbit_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 READ_FRO_FLAG, TIME_ORBIT_FLAG
      INTEGER*4 TIME_REF, READING_OSV_FLAG
      CHARACTER*LENGTH_NAME FILE_NAME
      REAL*8 START_RANGE, STOP_RANGE
      INTEGER*4 IERR(XD_NUM_ERR_READ_ORBIT_FILE), STATUS
      XD_ORBIT_FILE ORBIT_DATA
      STATUS = XD_READ_ORBIT_FILE (FILE_NAME, READ_FRO_FLAG,
     &                             TIME_ORBIT_FLAG, TIME_REF,
     &                             START_RANGE, STOP_RANGE,
     &                             READING_OSV_FLAG,
     &                             ORBIT_DATA, IERR)
```

7.3.3 Input parameters

The **xd_read_orbit_file** CFI function has the following input parameters:

Table 12: Input parameters of xd_read_orbit_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Orbit file name	-	-
read_fro_flag	long*	-	flag to indicate if the input file is: • a predicted orbit file • a restituted orbit file or a DORIS Preliminary file		• XD_TRUE for ROF and DORIS files • XD_FALSE for POF files
time_orbit_flag	long*	-	Flag for selecting the time range of the initialisation. Select either: • XD_SEL_FILE: for reading the whole file • XD_SEL_ORBIT: for reading the interval given by the start_range and the stop range parameters in orbits • XD_SEL_TIME: for reading the interval given by the start_range and the stop range parameters in days	-	All
time_ref	long*	-	Time reference if time_orbit_flag is XD_SEL_TIME. Dummy otherwise.	-	-
reading_osv_flag	long*	-	flag to indicate if the state vectors data have to be read.	-	• XD_TRUE for reading the state vector data • XD_FALSE for reading just the times and orbit numbers
start_range	double*	-	Start orbit or day	orbits or days	-
stop_range	double*	-	Stop orbit or day	orbits or days	-

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Time model ID: time_model. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].
- Time range initialisation flag: time_orbit_flag. See current document, section 6.2.

7.3.4 Output parameters

The output parameters of the **xd_read_orbit_file** CFI function are:

Table 13: Output parameters of xd_read_orbit_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_orbit_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
orbit_data	xd_orbit_file	-	Data structure containing the data read from the file	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *orbit_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_orbit_file**

7.3.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_orbit_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_orbit_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 14: Error messages of xd_read_orbit_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error in reading file	No calculation performed	XD_CFI_READ_ORBIT_FILE_READ_ERR	0
ERR	Error in getting the first element inside the input range	No calculation performed	XD_CFI_READ_ORBIT_FILE_INPUT_RANGE_ERR	1
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_ORBIT_FILE_MEMORY_ERR	2
ERR	Internal Error # 1	No calculation performed	XD_CFI_READ_ORBIT_FILE_INTERNAL_1_ERR	3
ERR	Error while reading data	No calculation performed	XD_CFI_READ_ORBIT_FILE_DATA_READ_ERR	4

Table 14: Error messages of xd_read_orbit_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Gap found after OSV no. %li	No calculation performed	XD_CFI_READ_ORBIT_FILE_GAP_ERR	5

7.3.6 Runtime performances

The following runtime performances have been measured.

Table 15: Runtime performances of xd_read_orbit_file function

Ultra Sparc II-400 [ms]
TBD

7.4 xd_free_orbit_file

7.4.1 Overview

The **xd_free_orbit_file** CFI function frees the memory allocated during the reading function **xd_read_orbit_file**.

7.4.2 Calling interface

The calling interface of the **xd_free_orbit_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_orbit_file orbit_data
      xd_free_orbit_file (&orbit_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_ORBIT_FILE ORBIT_DATA
      STATUS = XD_FREE_ORBIT_FILE (&ORBIT_DATA)
```

7.4.3 Input parameters

The **xd_free_orbit_file** CFI function has the following input parameters:

Table 16: Input parameters of xd_free_orbit_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_data	xd_orbit_file	-	Orbit data structure	-	-

7.4.4 Output parameters

This function does not return any value nor parameters.

7.5 xd_read_doris

7.5.1 Overview

The **xd_read_doris** CFI function reads DORIS Navigator files for Cryosat.

7.5.2 Calling interface

The calling interface of the **xd_read_doris** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
       long status, ;
       char *doris_file;
       long time_mode, interpol_flag;
       double time0, time1;
       xd_doris_file doris_data
       long ierr[XD_NUM_ERR_READ_DORIS];

       status = xd_read_doris(doris_file, &time_mode,
                              &time0, &time1,
                              &interpol_flag,
                              &doris_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

       INTEGER*4 TIME_MODE, INTERPOL_FLAG
       CHARACTER*LENGTH_NAME DORIS_FILE
       REAL*8 TIME0, TIME1
       INTEGER*4 IERR(XD_NUM_ERR_READ_DORIS), STATUS
       XD_DORIS_FILE DORIS_DATA

       STATUS = XD_READ_DORIS(DORIS_FILE, TIME_MODE,
&                             TIME0, TIME1, INTERPOL_FLAG,
&                             DORIS_DATA, IERR)
```

7.5.3 Input parameters

The **xd_read_doris** CFI function has the following input parameters:

Table 17: Input parameters of xd_read_doris function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
doris_file	char*	-	DORIS Navigator file name	-	-
time_mode	long	-	Flag for reading the whole file or just the requested time window	-	• XD_SEL_FILE or • XD_SEL_TIME
time0	double	-	Start time for the requested time window (if XD_SEL_TIME selected)	days in UTC	-
time1	double	-	Stop time for the requested time window (if XD_SEL_TIME selected)	days in UTC	-
interpol_flag	long	-	Flag to indicate if the read data are used for interpolation purposes. In that case 4 extra state vectors are read out of the requested time window	-	• XD_TRUE for interpol data • XD_FALSE otherwise

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Time model ID: time_mode. See [GEN_SUM].

7.5.4 Output parameters

The output parameters of the **xd_read_doris** CFI function are:

Table 18: Output parameters of xd_read_doris function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_doris	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
doris_data	xd_doris_file	-	DORIS data	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *doris_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_doris**.

7.5.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_doris** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_doris** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 19: Error messages of xd_read_doris function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error in memory assignation	No calculation performed	XD_CFI_READ_DORIS_ERROR_IN_MEMORY_ASIG_ERR	0
ERR	Wrong input parameter value: \"time_mode\"	No calculation performed	XD_CFI_READ_DORIS_WRONG_TIME_MODE_ERR	1
ERR	Wrong time on input (start time after stop time)	No calculation performed	XD_CFI_READ_DORIS_WRONG_TIME_1_ERR	2
ERR	Wrong time on input (out of limits)	No calculation performed	XD_RCFI_EAD_DORIS_WRONG_TIME_2_ERR	3
ERR	DORIS level 0 filename not supplied	No calculation performed	XD_CFI_READ_DORIS_NO_FILENAME_ERR	4
ERR	DORIS Level 0 file cannot be open	No calculation performed	XD_CFI_READ_DORIS_CANNOT_OPEN_ERR	5
ERR	Could not find keyword: %s	No calculation performed	XD_CFI_READ_DORIS_FINDKW_ERROR_ERR	6
ERR	Error reading DORIS data for keyword: %s	No calculation performed	XD_CFI_READ_DORIS_READ_ERR	7
ERR	Error reading DORIS binary data	No calculation performed	XD_CFI_READ_DORIS_READ_BIN_ERR	8
ERR	Error changing time from ascii to processing	No calculation performed	XD_CFI_READ_DORIS_ASCII_TO_PROCESSING_ERR	9
ERR	Gap found reading DORIS level0 data	No calculation performed	XD_CFI_READ_DORIS_GAP_IN_FILE_ERR	10
ERR	DORIS file does not cover user required time in terval	No calculation performed	XD_CFI_READ_DORIS_DOES_NOT_COVER_TIME_INTERVAL_ERR	11

7.5.6 Runtime performances

The following runtime performances have been measured.

Table 20: Runtime performances of xd_read_doris function

Ultra Sparc II-400 [ms]
TBD

7.6 xd_free_doris

7.6.1 Overview

The **xd_free_doris** CFI function frees the memory allocated during the reading function **xd_read_doris**.

7.6.2 Calling interface

The calling interface of the **xd_free_doris** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_doris_file doris_data
      xd_free_doris (&doris_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_DORIS_FILE DORIS_DATA
      XD_FREE_DORIS (DORIS_DATA)
```

7.6.3 Input parameters

The **xd_free_doris** CFI function has the following input parameters:

Table 21: Input parameters of xd_free_doris function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
doris_data	xd_doris_file	-	DORIS data structure	-	-

7.6.4 Output parameters

This function does not return any value nor parameters.

7.7 xd_read_doris_header

7.7.1 Overview

The **xd_read_doris_header** CFI function reads the Main Product Header (MPH) and the Specific Product Header (SPH) from DORIS Navigator files for Cryosat.

7.7.2 Calling interface

The calling interface of the **xd_read_doris_header** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *doris_file;
      xd_doris_mph_sph doris_hdr;
      long ierr[XD_NUM_ERR_READ_DORIS_HEADER];

      status = xd_read_doris_header(doris_file, &doris_hdr, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME DORIS_FILE
      INTEGER*4 IERR(XD_NUM_ERR_READ_DORIS_HEADER), STATUS
      XD_DORIS_MPH_SPH DORIS_HDR

      STATUS = XD_READ_DORIS_HEADER(DORIS_FILE, DORIS_DATA, IERR)
```

7.7.3 Input parameters

The **xd_read_doris_header** CFI function has the following input parameters:

Table 22: Input parameters of xd_read_doris_header function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
doris_file	char*	-	DORIS file name	-	-

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Time model ID: time_mode. See [GEN_SUM].

7.7.4 Output parameters

The output parameters of the **xd_read_doris_header** CFI function are:

Table 23: Output parameters of xd_read_doris_header function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_doris_header	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
doris_data	xd_doris_mph_sph	-	doris header structure	-	-
ierr	long []	.	Error vector	-	-

7.7.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_doris_header** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_doris_header** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 24: Error messages of xd_read_doris function

Error type	Error message	Cause and impact	Error code	Error No
ERR	DORIS level 0 filename not supplied	No calculation performed	XD_CFI_READ_DORIS_HEADER_NO_FILENAME_ERR	0
ERR	DORIS Level 0 file cannot be open	No calculation performed	XD_CFI_READ_DORIS_HEADER_CANNOT_OPEN_ERR	1
ERR	Could not find keyword: %s	No calculation performed	XD_CFI_READ_DORIS_HEADER_FINDKW_ERROR_ERR	2
ERR	Error reading DORIS data for keyword: %s	No calculation performed	XD_CFI_READ_DORIS_HEADER_READ_ERR	3

7.7.6 Runtime performances

The following runtime performances have been measured.

Table 25: Runtime performances of xd_read_doris_header function

Ultra Sparc II-400 [ms]
TBD

7.8 xd_read_osf

7.8.1 Overview

The **xd_read_osf** CFI function reads Orbit Scenario files for Earth Explorer Missions. The files have to be written in XML and consits on a list of orbital changes of the satellite along the orbit.

This function can also be used for reading the list of orbital changes within Orbit Event files.

7.8.2 Calling interface

The calling interface of the **xd_read_osf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_osf_file osf_data;
      long ierr[XD_NUM_ERR_READ_OSF];

      status = xd_read_osf (file_name, &osf_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>
      INTEGER*4 STATUS
      CHARACTER*LENGTH_NAME FILE_NAME
      XD_OSF_FILE OSF_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_OSF), STATUS

      STATUS = XD_READ_OSF(FILE_NAME, OSF_DATA, IERR)
```

7.8.3 Input parameters

The **xd_read_osf** CFI function has the following input parameters:

Table 26: Input parameters of xd_read_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Orbit Scenario file name	-	-

7.8.4 Output parameters

The output parameters of the **xd_read_osf** CFI function are:

Table 27: Output parameters of xd_read_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_osf	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
osf_data	xd_osf_file	-	Structure with the OSF data	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *osf_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_osf**.

7.8.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_osf** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_osf** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 28: Error messages of xd_read_osf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error initializing the file parser	No calculation performed	XD_CFI_READ_XML_OSF_INIT_PARSER_ERR	0
ERR	Error finding the data block keyword	No calculation performed	XD_CFI_READ_XML_OSF_XML_DATA_BLOCK_ERR	1
ERR	Error reading the data block attribute	No calculation performed	XD_CFI_READ_XML_OSF_XML_ATTRIBUTE_ERR	2
ERR	"Error reading the xml attribute"	No calculation performed	XD_CFI_READ_XML_OSF_XML_TYPE_ERR	3
ERR	Error reading XML element: %s	No calculation performed	XD_CFI_READ_XML_OSF_READ_PARAM_ERR	4
ERR	Error the size of the list (negative)	No calculation performed	XD_CFI_READ_XML_OSF_XML_DATA_BLOCK_SIZE_ERR	5
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_XML_OSF_MEMORY_ERR	6

7.8.6 Runtime performances

The following runtime performances have been measured.

Table 29: Runtime performances of xd_read_osf function

Ultra Sparc II-400 [ms]
TBD

7.9 xd_free_osf

7.9.1 Overview

The **xd_free_osf** CFI function frees the memory allocated during the reading function **xd_read_osf**.

7.9.2 Calling interface

The calling interface of the **xd_free_osf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_osf_file osf_data
      xd_free_osf (&osf_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_OSF_FILE OSF_DATA
      XD_FREE_OSF (OSF_DATA)
```

7.9.3 Input parameters

The **xd_free_osf** CFI function has the following input parameters:

Table 30: Input parameters of xd_free_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
osf_data	xd_osf_file	-	DORIS data structure	-	-

7.9.4 Output parameters

This function does not return any value nor parameters.

7.10 xd_read_sdf

7.10.1 Overview

The **xd_read_sdf** CFI function reads Swath Definition files for Earth Explorer Missions.

7.10.2 Calling interface

The calling interface of the **xd_read_sdf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      xd_sdf_file sdf_data;
      char *file_name;
      long ierr[XD_NUM_ERR_READ_SDF];

      status = xd_read_sdf (file_name, &sdf_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_SDF_FILE SDF_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_SDF), STATUS

      STATUS = XD_READ_SDF(FILE_NAME, SDF_DATA,IERR)
```

7.10.3 Input parameters

The **xd_read_sdf** CFI function has the following input parameters:

Table 31: Input parameters of xd_read_sdf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Swath Definition file name	-	-

7.10.4 Output parameters

The output parameters of the **xd_read_sdf** CFI function are:

Table 32: Output parameters of xd_read_sdf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_sdf	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
sdf_data	xd_sdf_file	-	Swath Definition data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *sdf_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_sdf**.

7.10.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_sdf** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_sdf** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 33: Error messages of xd_read_sdf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error opening Swath Definition file	No calculation performed	XD_CFI_READ_SDF_OPEN_FILE_ERR	0
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_SDF_MEMORY_ERR	1
ERR	Error reading swath record %d	No calculation performed	XD_CFI_READ_SDF_REC_READ_ERR	2

7.10.6 Runtime performances

The following runtime performances have been measured.

Table 34: Runtime performances of xd_read_sdf function

Ultra Sparc II-400 [ms]
TBD

7.11 xd_free_sdf

7.11.1 Overview

The **xd_free_sdf** CFI function frees the memory allocated during the reading function **xd_read_sdf**.

7.11.2 Calling interface

The calling interface of the **xd_free_sdf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_sdf_file sdf_data
      xd_free_sdf (&sdf_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_SDF_FILE SDF_DATA
      XD_FREE_sdf (SDF_DATA)
```

7.11.3 Input parameters

The **xd_free_sdf** CFI function has the following input parameters:

Table 35: Input parameters of xd_free_sdf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sdf_data	xd_sdf_file	-	SDF data structure	-	-

7.11.4 Output parameters

This function does not return any value nor parameters.

7.12 xd_read_stf

7.12.1 Overview

The **xd_read_stf** CFI function reads Swath Template Files for Earth Explorer Missions.

7.12.2 Calling interface

The calling interface of the **xd_read_stf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_stf_file stf_data;
      long ierr[XD_NUM_ERR_READ_STF];

      status = xd_read_stf (file_name, &stf_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 STATUS
      CHARACTER*LENGTH_NAME FILE_NAME
      XD_STF_FILE STF_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_STF), STATUS

      STATUS = XD_READ_STF(FILE_NAME, STF_DATA,IERR)
```

7.12.3 Input parameters

The **xd_read_stf** CFI function has the following input parameters:

Table 36: Input parameters of xd_read_stf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*^	-	Swath Template file name	-	-

7.12.4 Output parameters

The output parameters of the **xd_read_stf** CFI function are:

Table 37: Output parameters of xd_read_stf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_stf	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
stf_data	xd_stf_file	-	Swath template file data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *stf_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_stf**.

7.12.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_stf** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_stf** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 38: Error messages of xd_read_stf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error initializing parser to read the file	No calculation performed	XD_CFI_READ_STF_INIT_PARSER_ERR	0
ERR	Error reading the variable header	No calculation performed	XD_READ_STF_VHR_ERR	1
ERR	Error reading element: %s"	No calculation performed	XD_CFI_READ_STF_PARAM_READ_ERR	2
ERR	Could not find data block.	No calculation performed	XD_CFI_READ_STF_DATA_BLOCK_ERR	3
ERR	Could not read Data_Block attribute.	No calculation performed	XD_CFI_READ_STF_ATTRIBUTE_ERR	4
ERR	Data block is not XML type.	No calculation performed	XD_CFI_READ_STF_XML_TYPE_ERR	5
ERR	egative number of swath coordinates	No calculation performed	XD_CFI_READ_STF_DATA_BLOCK_SIZE_ERR	6
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_STF_MEMORY_ERR	7
ERR	Error reading swath point #%d	No calculation performed	XD_CFI_READ_STF_SWATH_POINT_READ_ERR	8
ERR	Error in STF, latitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_LAT_ERR	9
ERR	Error in STF, longitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_LONG_ERR	10
ERR	Error in STF, left latitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_LEFT_LAT_ERR	11
ERR	Error in STF, middle latitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_MID_LAT_ERR	12
ERR	Error in STF, right latitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_RIGHT_LAT_ERR	13
ERR	Error in STF, left longitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_LEFT_LONG_ERR	14
ERR	Error in STF, middle longitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_MID_LONG_ERR	15
ERR	Error in STF, right longitude record out of range for swath point #%d	No calculation performed	XD_CFI_READ_STF_WRONG_RIGHT_LONG_ERR	16

7.12.6 Runtime performances

The following runtime performances have been measured.

Table 39: Runtime performances of xd_read_stf function

Ultra Sparc II-400 [ms]
TBD

7.13 xd_free_stf

7.13.1 Overview

The **xd_free_stf** CFI function frees the memory allocated during the reading function **xd_read_stf**.

7.13.2 Calling interface

The calling interface of the **xd_free_stf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_stf_file stf_data
      xd_free_stf (&stf_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_STF_FILE STF_DATA
      XD_FREE_stf (STF_DATA)
```

7.13.3 Input parameters

The **xd_free_stf** CFI function has the following input parameters:

Table 40: Input parameters of xd_free_stf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
stf_data	xd_stf_file	-	STF data structure	-	-

7.13.4 Output parameters

This function does not return any value nor parameters.

7.14 xd_read_stf_vhr

7.14.1 Overview

The **xd_read_stf_vhr** CFI function reads the varible header in Swath Template File for Earth Explorer Missions.

7.14.2 Calling interface

The calling interface of the **xd_read_stf_vhr** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_stf_vhr vhr_data;
      long ierr[XD_NUM_ERR_READ_STF_VHR];

      status = xd_read_stf_vhr (file_name, &vhr_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 STATUS
      CHARACTER*LENGTH_NAME FILE_NAME
      XD_STF_VHR VHR_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_STF_VHR), STATUS

      STATUS = XD_READ_STF_VHR(FILE_NAME, VHR_DATA,IERR)
```

7.14.3 Input parameters

The **xd_read_stf_vhr** CFI function has the following input parameters:

Table 41: Input parameters of xd_read_stf_vhr function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*^	-	Swath Template file name	-	-

7.14.4 Output parameters

The output parameters of the **xd_read_stf_vhr** CFI function are:

Table 42: Output parameters of xd_read_stf_vhr function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_stf_vhr	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
vhr_data	xd_stf_vhr	-	Data structure for the Swath template variable header	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *vhr_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_stf_vhr**.

7.14.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_stf_vhr** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_stf_vhr** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 43: Error messages of xd_read_stf_vhr function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error initializing parser to read the file	No calculation performed	XD_CFI_READ_STF_VHR_INIT_PARSER_ERR	0
ERR	Could not find variable header	No calculation performed	XD_CFI_READ_STF_VHR_VARIABLE_HEADER_ERR	1
ERR	Error within the reading function	No calculation performed	XD_CFI_READ_STF_VHR_INTERNAL_1_ERR	2
ERR	Error reading element: %s	No calculation performed	XD_CFI_READ_STF_VHR_PARAM_READ_ERR	3
ERR	Incorrect swath type	No calculation performed	XD_CFI_READ_STF_VHR_SWATH_TYPE_ERR	4
ERR	Error reading "Orbit_State_Vector"	No calculation performed	XD_CFI_READ_STF_VHR_ORBIT_PARAMS_ERR	5
ERR	Error reading "Orbit_Geometry"	No calculation performed	XD_CFI_READ_STF_VHR_GEOM_PARAMS_ERR	6
ERR	Error reading altitude	No calculation performed	XD_CFI_READ_STF_VHR_ALTITUDE_READ_ERR	7
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_STF_VHR_MEMORY_ERR	8

7.14.6 Runtime performances

The following runtime performances have been measured.

Table 44: Runtime performances of xd_read_stf_vhr function

Ultra Sparc II-400 [ms]
TBD

7.15 xd_free_stf_vhr

7.15.1 Overview

The **xd_free_stf_vhr** CFI function frees the memory allocated during the reading function **xd_read_stf_vhr**.

7.15.2 Calling interface

The calling interface of the **xd_free_stf_vhr** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_stf_vhr stf_vhr;
      xd_free_stf_vhr (&stf_vhr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_STF_VHR STF_VHR
      XD_FREE_STF_VHR (STF_VHR)
```

7.15.3 Input parameters

The **xd_free_stf_vhr** CFI function has the following input parameters:

Table 45: Input parameters of xd_free_stf_vhr function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
stf_vhr	xd_stf_vhr	-	STF variable header data structure	-	-

7.15.4 Output parameters

This function does not return any value nor parameters.

7.16 xd_read_att

7.16.1 Overview

The xd_read_att CFI function reads attitude generic files. This files have to be written in XML and consits on a list of attitude angles or quaternions.

7.16.2 Calling interface

The calling interface of the **xd_read_att** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      xd_att_file att_data;
      char *file_name;
      long ierr[XD_NUM_ERR_READ_ATT];

      status = xd_read_att (file_name, att_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the #include statement):

```
#include <explorer_data_handling.inc>

      XD_ATT_FILE ATT_DATA
      CHARACTER*LENGTH_NAME FILE_NAME
      INTEGER*4 IERR(XD_NUM_ERR_READ_ATT), STATUS

      STATUS = XD_READ_ATT(FILE_NAME, ATT_DATA, IERR)
```

7.16.3 Input parameters

The **xd_read_att** CFI function has the following input parameters:

Table 46: Input parameters of xd_read_att function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Attitude file name	-	-

7.16.4 Output parameters

The output parameters of the **xd_read_** CFI function are:

Table 47: Output parameters of xd_read_att function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_att	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
att_data	xd_att_file	-	Attitude data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *att_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_att**.

7.16.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_att** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_att** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 48: Error messages of xd_read_att function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error initializing parser to read the file	No calculation performed	XD_CFI_READ_ATT_INIT_PARSER_ERR	0
ERR	Error reading element: %s	No calculation performed	XD_CFI_READ_ATT_READ_PARAM_ERR	1
ERR	Wrong file type	No calculation performed	XD_CFI_READ_ATT_WRONG_FILE_TYPE_ERR	2
ERR	Error navigating through the file	No calculation performed	XD_CFI_READ_XML_ATT_NAVIGATION_ERR	3
ERR	Wrong attitude data type. Only "Quaternions" and "Attitude_Angles_Data" allowed	No calculation performed	XD_CFI_READ_ATT_WRONG_DATA_TYPE_ERR	4
ERR	Inconsistent values for <Attitude_Data_Type> and the list of attitude data	No calculation performed	XD_CFI_READ_ATT_INCONSISTENT_DATA_TYPE_ERR	5
ERR	Wrong number of records in the list	No calculation performed	XD_CFI_READ_ATT_XML_DATA_BLOCK_SIZE_ERR	6
ERR	Wrong parameter in "Inertial_Ref_Frame"	No calculation performed	XD_CFI_READ_ATT_WRONG_REF_FRAME_ERR	7
ERR	Error reading attitude data list	No calculation performed	XD_CFI_READ_ATT_READ_LIST_ERR	8
ERR	Error converting ascii date to processing	No calculation performed	XD_CFI_READ_ATT_TIME_CONV_ERR	9
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_ATT_MEMORY_ERR	10
ERR	Could not close the file	No calculation performed	XD_CFI_READ_ATT_CLEANUP_PARSER_ERR	11
ERR	Wrong time reference for element n. %d. All time references should be equal	No calculation performed	XD_CFI_READ_ATT_WRONG_TIME_REF_ERR	12

Table 48: Error messages of xd_read_att function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Quaternion modulus out of limits. Check list element n. %d	No calculation performed	XD_CFI_READ_ATT_WRONG_QUATERNION_ERR	13
ERR	Angle out of limits. Check list element n. %d	No calculation performed	XD_CFI_READ_ATT_WRONG_ANGLE_ERR	14
ERR	Maximum Gap value must be positive	No calculation performed	XD_CFI_READ_ATT_MAX_GAP_ERR	15

7.16.6 Runtime performances

The following runtime performances have been measured.

Table 49: Runtime performances of xd_read_att function

Ultra Sparc II-400 [ms]
TBD

7.17 xd_free_att

7.17.1 Overview

The **xd_free_att** CFI function frees the memory allocated during the reading function **xd_read_att**.

7.17.2 Calling interface

The calling interface of the **xd_free_att** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_att_file att_data;
      xd_free_att (&att_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_ATT_FILE ATT_DATA
      XD_FREE_ATT (ATT_DATA)
```

7.17.3 Input parameters

The **xd_free_att** CFI function has the following input parameters:

Table 50: Input parameters of xd_free_att function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
att_data	xd_att_file	-	Attitude data structure	-	-

7.17.4 Output parameters

This function does not return any value nor parameters.

7.18 xd_read_star_tracker

7.18.1 Overview

The **xd_read_star_tracker** CFI function reads a list of star tracker files for Cryosat.

7.18.2 Calling interface

The calling interface of the **xd_read_star_tracker** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      long n_files, time_init_mode;
      char **file_list;
      double time0, time1;
      xd_tracker_limits str_limit;
      xd_star_tracker_file str_data;
      long ierr[XD_NUM_ERR_READ_STAR_TRACKER];

      status = xd_read_star_tracker (&n_files, file_list,
                                     &time_init_mode, &time0, &time1,
                                     &str_limit,
                                     &str_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 N_FILEs, TIME_INIT_MODE
      CHARACTER*LENGTH_NAME FILE_LIST(N_FILE)
      REAL*8 TIME0, TIME1
      XD_TRACKER_LIMITS STR_LIMIT
      XD_STAR_TRACKER_FILE STR_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_STAR_TRACKER), STATUS

      STATUS = XD_READ_STAR_TRACKER(N_FILES, FILE_LIST,
&                                   TIME_INIT_MODE, TIME0, TIME1,
&                                   STR_LIMIT,
&                                   STR_DATA, IERR)
```

7.18.3 Input parameters

The **xd_read_star_tracker** CFI function has the following input parameters:

Table 51: Input parameters of xd_read_star_tracker function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
n_files	long	-	Number of input files	-	> 0
file_list	char **	-	List of star tracker files	-	-
time_init_mode	long	-	Flag for reading the whole file or just the requested time window		• XD_SEL_FILE or • XD_SEL_TIME
time0	double	-	Start time for the requested time window	-	days (TAI)
time1	double	-	Stop time for the requested time window	-	days (TAI)
str_limit	xd_str_limits	-	data structure containing the limits for the quaternion validation	-	-

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Time range initialisation flag: time_init_mode. See current document, section 6.2.

7.18.4 Output parameters

The output parameters of the **xd_read_star_tracker** CFI function are:

Table 52: Output parameters of xd_read_star_tracker function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_star_tracker	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
str_data	xd_star_tracker_file	-	Star tracker data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *str_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_star_tracker**.

7.18.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_star_tracker** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_star_tracker** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 53: Error messages of xd_read_star_tracker function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open input file	No calculation performed	XD_CFI_READ_STR_TRACKER_OPEN_FILE_ERR	0
ERR	Could not read input file	No calculation performed	XD_CFI_READ_STR_TRACKER_READ_FILE_ERR	1
ERR	Memory allocation error	No calculation performed	XD_CFI_READ_STR_TRACKER_MEMORY_FILE_ERR	2
ERR	Gap between quaternions above maximum allowed value after time %f	No calculation performed	XD_CFI_READ_STR_TRACKER_GAP_ERR	3
ERR	No enough valid quaternions to cover the requested interval	No calculation performed	XD_CFI_READ_STR_TRACKER_NO_ENOUGH_DATA_ERR	4

7.18.6 Runtime performances

The following runtime performances have been measured.

Table 54: Runtime performances of xd_read_star_tracker function

Ultra Sparc II-400 [ms]
TBD

7.19 xd_free_star_tracker

7.19.1 Overview

The **xd_free_star_tracker** CFI function frees the memory allocated during the reading function **xd_read_star_tracker**.

7.19.2 Calling interface

The calling interface of the **xd_free_star_tracker** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_star_tracker_file str_data;
      xd_free_star_tracker (&str_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_STAR_TRACKER_FILE STR_DATA
      XD_FREE_STAR_TRACKER (STR_DATA)
```

7.19.3 Input parameters

The **xd_free_star_tracker** CFI function has the following input parameters:

Table 55: Input parameters of xd_free_star_tracker function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
str_data	xd_star_tracker_file	-	Star tracker data structure	-	-

7.19.4 Output parameters

This function does not return any value nor parameters.

7.20 xd_read_star_tracker_conf_file

7.20.1 Overview

The **xd_read_star_tracker_conf_file** CFI function reads an star tracker configuration file for Cryosat. The files have to be written in XML.

7.20.2 Calling interface

The calling interface of the **xd_read_star_tracker_conf_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status, star_tracker_id;
      char *file_name;
      xd_tracker_conf_file conf_data;
      long ierr[XD_NUM_ERR_READ_STAR_TRACKER_CONF_FILE];

      status = xd_read_star_tracker_conf_file (file_name,
                                               &star_tracker_id,
                                               &conf_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 STATUS, STAR_TRACKER_ID
      CHARACTER*LENGTH_NAME FILE_NAME
      XD_TRACKER_CONF_FILE CONF_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_TRACKER_CONF_FILE), STATUS
      STATUS = XD_READ_STAR_TRACKER_CONF_FILE(FILE_NAME,
&                                             STAR_TRACKER_ID,
&                                             CONF_DATA, IERR)
```

7.20.3 Input parameters

The **xd_read_star_tracker_conf_file** CFI function has the following input parameters:

Table 56: Input parameters of xd_read_star_tracker_conf_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Star Tracker configuration file name	-	-
star_tracker_id	long	-	Star tracker number for which the configuration data is to be read	-	1, 2 or 3

7.20.4 Output parameters

The output parameters of the **xd_read_star_tracker_conf_file** CFI function are:

Table 57: Output parameters of xd_read_star_tracker_conf_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_star_tracker_conf_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
conf_data	xd_tracker_conf_file	-	Star tracker configuration data structure with	-	-
ierr	long[]	-	Error vector	-	-

7.20.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_star_tracker_conf_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_star_tracker_conf_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 58: Error messages of xd_read_star_tracker_conf_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input file	No calculation performed	XD_CFI_READ_STR_CONF_FILE_READ_FILE_ERR	0

7.20.6 Runtime performances

The following runtime performances have been measured.

Table 59: Runtime performances of xd_read_star_tracker_conf_file function

Ultra Sparc II-400 [ms]
TBD

7.21 xd_read_dem

7.21.1 Overview

The **xd_read_dem** CFI function reads a DEM file providing the table with the altitudes for each point of the grid of the DEM file.

7.21.2 Calling interface

The calling interface of the **xd_read_dem** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *dem_name;
      xd_dem_config_file dem_conf_data;
      xd_dem_file dem_data;
      long ierr[XD_NUM_ERR_READ_DEM];

      status = xd_read_dem (dem_name, &dem_conf_data,
                            &dem_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME DEM_NAME
      XD_DEM_CONFIG_FILE DEM_CONF_DATA
      XD_DEM_FILE DEM_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_DEM), STATUS

      STATUS = XD_READ_DEM(DEM_NAME, DEM_CONF_DATA,
                           DEM_DATA, IERR)
```

7.21.3 Input parameters

The **xd_read_dem** CFI function has the following input parameters:

Table 60: Input parameters of xd_read_dem function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
dem_name	char*	-	DEM file name (do not include the path)	-	-

Table 60: Input parameters of xd_read_dem function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
dem_conf_data	xd_dem_config_file	-	DEM configuration data structure. This data are read from a configuration file with **xd_read_dem_config_file**	-	-

It is possible to use enumeration values rather than integer values for some of the input arguments:

- Time model ID: time_model. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].
- Time range initialisation flag: time_init_mode. See current document, section 6.2.

7.21.4 Output parameters

The output parameters of the **xd_read_dem** CFI function are:

Table 61: Output parameters of xd_read_dem function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_dem	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
dem_data	xd_dem_file	-	DEM data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *dem_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_dem**.

7.21.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_dem** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_dem** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 62: Error messages of xd_read_dem function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Memory allocation error	No calculation performed	XD_CFI_READ_DEM_MEMORY_ERR	0
ERR	Incorrect input DEM configuration file	No calculation performed	XD_CFI_READ_DEM_NO_CONFIG_FILE_ERR	1
ERR	Wrong input file name	No calculation performed	XD_CFI_READ_DEM_WRONG_FILENAME_ERR	2
ERR	Could not open the DEM file	No calculation performed	XD_CFI_READ_DEM_OPEN_FILE_ERR	3
ERR	Could not read the DEM file	No calculation performed	XD_CFI_READ_DEM_READ_FILE_ERR	5

7.21.6 Runtime performances

The following runtime performances have been measured.

Table 63: Runtime performances of xd_read_dem function

Ultra Sparc II-400 [ms]
TBD

7.22 xd_free_dem

7.22.1 Overview

The **xd_free_dem** CFI function frees the memory allocated in the reading function **xd_read_dem**.

7.22.2 Calling interface

The calling interface of the **xd_free_dem** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_dem_file dem_data;
      xd_free_dem (&dem_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_DEM_FILE DEM_DATA
      XD_FREE_DEM (DEM_DATA)
```

7.22.3 Input parameters

The **xd_free_dem** CFI function has the following input parameters:

Table 64: Input parameters of xd_free_dem function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
dem_data	xd_dem_file	-	DEM data structure	-	-

7.22.4 Output parameters

This function does not return any value nor parameters.

7.23 xd_read_dem_config_file

7.23.1 Overview

The **xd_read_dem_config_file** CFI function reads for Earth Explorer Missions.

7.23.2 Calling interface

The calling interface of the **xd_read_dem_config_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
       long status;
       char *file_name;
       xd_dem_config_file dem_config_data;
       long ierr[XD_NUM_ERR_READ_DEM_CONFIG];

       status = xd_read_dem_config_file (file_name,
                                         &dem_config_data,
                                         ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

       INTEGER*4 STATUS
       CHARACTER*LENGTH_NAME FILE_NAME
       XD_DEM_CONFIG_FILE DEM_CONFIG_DATA
       INTEGER*4 IERR(XD_NUM_ERR_READ_DEM_CONFIG_FILE), STATUS

       STATUS = XD_READ_DEM_CONFIG_FILE(FILE_NAME, DEM_CONFIG_DATA,IERR)
```

7.23.3 Input parameters

The **xd_read_dem_config_file** CFI function has the following input parameters:

Table 65: Input parameters of xd_read_dem_config_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	DEM configuration file name	-	-

7.23.4 Output parameters

The output parameters of the **xd_read_dem_config_file** CFI function are:

Table 66: Output parameters of xd_read_dem_config_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_dem_config_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
dem_config_data	xd_dem_config_file	-	DEM configuration data structure	-	-
ierr	long[]	-	Error vector	-	-

7.23.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_dem_config_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_dem_config_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 67: Error messages of xd_read_dem_config_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open the configuration file	No calculation performed	XD_CFI_READ_DEM_CONFIG_FILE_OPEN_ERR	0
ERR	Could not read the configuration file	No calculation performed	XD_CFI_READ_DEM_CONFIG_FILE_READ_ERR	1
ERR	Could not open the model tag	No calculation performed	XD_CFI_READ_DEM_CONFIG_FILE_READ_MODEL_ERR	2
ERR	Memory allocation error	No calculation performed	XD_CFI_READ_DEM_CONFIG_FILE_MEMORY_ERR	3
ERR	Could not open a ACE file	No calculation performed	XD_CFI_READ_DEM_CONFIG_FILE_OPEN_DEM_FILE_ERR	4

Table 67: Error messages of xd_read_dem_config_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not read a ACE file	No calculation performed	XD_CFI_READ_DEM_CONFIG_FILE_READ_DEM_FILE_ERR	5

7.23.6 Runtime performances

The following runtime performances have been measured.

Table 68: Runtime performances of xd_read_dem_config_file function

Ultra Sparc II-400 [ms]
TBD

7.24 xd_read_zone

7.24.1 Overview

The **xd_read_zone** CFI function reads a specific zone from a zone database file for Earth Explorer Missions.

7.24.2 Calling interface

The calling interface of the **xd_read_zone** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *zone_id;
      char *file_name;
      xd_zone_rec zone_rec;
      long ierr[XD_NUM_ERR_READ_ZONE];

      status = xd_read_zone (file_name, &zone_id, &zone_rec, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      CHARACTER*8 ZONE_ID
      XD_ZONE_REC ZONE_REC
      INTEGER*4 IERR(XD_NUM_ERR_READ_ZONE), STATUS

      STATUS = XD_READ_ZONE_FILE(FILE_NAME, ZONE_ID, ZONE_REC, IERR)
```

7.24.3 Input parameters

The **xd_read_zone** CFI function has the following input parameters:

Table 69: Input parameters of xd_read_zone function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Zone database file name	-	-
zone_id	char*	-	Zone Id to be read	-	-

7.24.4 Output parameters

The output parameters of the **xd_read_zone** CFI function are:

Table 70: Output parameters of xd_read_zone function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_zone	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
zone_rec	xd_zone_rec	-	Zone Data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *zone_rec* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_zone**.

7.24.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_zone** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_zone** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 71: Error messages of xd_read_zone function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Zone File not found	No calculation performed	XD_CFI_READ_ZONE_INIT_PARSER_ERR	0
ERR	Data Block not found	No calculation performed	XD_CFI_READ_ZONE_DATA_BLOCK_ERR	1
ERR	Data Block attribute not read	No calculation performed	XD_CFI_READ_ZONE_DATA_BLOCK_ATTRIBUTE_ERR	2

Table 71: Error messages of xd_read_zone function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Data Block not of XML type	No calculation performed	XD_CFI_READ_ZONE_XML_TYPE_ERR	3
ERR	List_of_Zones not found.	No calculation performed	XD_CFI_READ_ZONE_LIST_ZONES_READ_ERR	4
ERR	List_of_Zones attribute not read.	No calculation performed	XD_CFI_READ_ZONE_LIST_ZONES_SIZE_ERR	5
ERR	Internal error returned	No calculation performed	XD_CFI_READ_ZONE_INTERNAL_1_ERR	6
ERR	Zone_ID cannot be read.	No calculation performed	XD_CFI_READ_ZONE_ZONE_ID_READ_ERR	7
ERR	Zone_ID not found.	No calculation performed	XD_CFI_READ_ZONE_ZONE_ID_NOT_FOUND_ERR	8
ERR	Error reading zone record	No calculation performed	XD_CFI_READ_ZONE_RECORD_READ_ERR	9

7.24.6 Runtime performances

The following runtime performances have been measured.

Table 72: Runtime performances of xd_read_zone function

Ultra Sparc II-400 [ms]
TBD

7.25 xd_free_zone

7.25.1 Overview

The **xd_free_zone** CFI function frees the memory allocated during the reading function **xd_read_zone**.

7.25.2 Calling interface

The calling interface of the **xd_free_zone** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_zone_rec zone_data;
      xd_free_zone (&zone_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_ZONE_REC ZONE_DATA
      XD_FREE_ZONE (ZONE_DATA)
```

7.25.3 Input parameters

The **xd_free_zone** CFI function has the following input parameters:

Table 73: Input parameters of xd_free_zone function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
zone_data	xd_zone_rec	-	Zone record data structure	-	-

7.25.4 Output parameters

This function does not return any value nor parameters.

7.26 xd_read_zone_file

7.26.1 Overview

The **xd_read_zone_file** CFI function reads a zone database file for Earth Explorer Missions.

7.26.2 Calling interface

The calling interface of the **xd_read_zone_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_zone_file zone_data;
      long ierr[XD_NUM_ERR_READ_ZONE_FILE];

      status = xd_read_zone_file (file_name, &zone_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_ZONE_FILE ZONE_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_ZONE_FILE), STATUS

      STATUS = XD_READ_ZONE_FILE(FILE_NAME, ZONE_DATA, IERR)
```

7.26.3 Input parameters

The **xd_read_zone_file** CFI function has the following input parameters:

Table 74: Input parameters of xd_read_zone_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Zone database file name	-	-

7.26.4 Output parameters

The output parameters of the **xd_read_zone_file** CFI function are:

Table 75: Output parameters of xd_read_zone_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_zone_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
xd_zone_file	zone_data	-	Structure containing the data for all the zones read from the file	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *zone_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_zone_file**.

7.26.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_zone_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_zone_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 76: Error messages of xd_read_zone_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Zone File not found.	No calculation performed	XD_CFI_READ_ZONE_FILE_INIT_PARSER_ERR	0
ERR	Data Block not found	No calculation performed	XD_CFI_READ_ZONE_FILE_DATA_BLOCK_ERR	1
ERR	Data Block attribute not read.	No calculation performed	XD_CFI_READ_ZONE_FILE_DATA_BLOCK_ATTRIBUTE_ERR	2
ERR	Data Block not of XML type.	No calculation performed	XD_CFI_READ_ZONE_FILE_XML_TYPE_ERR	3

Table 76: Error messages of xd_read_zone_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	List_of_Zones not found.	No calculation performed	XD_CFI_READ_ZONE_FILE_LIST_ZONES_READ_ERR	4
ERR	List_of_Zones attribute not read	No calculation performed	XD_CFI_READ_ZONE_FILE_LIST_ZONES_SIZE_ERR	5
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_ZONE_FILE_MEM_ERR	6
ERR	Error reading zone record number %d	No calculation performed	XD_CFI_READ_ZONE_FILE_RECORD_READ_ERR	7

7.26.6 Runtime performances

The following runtime performances have been measured.

Table 77: Runtime performances of xd_read_zone_file function

Ultra Sparc II-400 [ms]
TBD

7.27 xd_free_zone_file

7.27.1 Overview

The **xd_free_zone_file** CFI function frees the memory allocated during the reading function **xd_read_zone_file**.

7.27.2 Calling interface

The calling interface of the **xd_free_zone_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_zone_file zone_data;
      xd_free_zone_file (&zone_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_ZONE_FILE ZONE_DATA
      XD_FREE_ZONE_FILE (ZONE_DATA)
```

7.27.3 Input parameters

The **xd_free_zone_file** CFI function has the following input parameters:

Table 78: Input parameters of xd_free_zone_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
zone_data	xd_zone_file	-	Zone file data structure	-	-

7.27.4 Output parameters

This function does not return any value nor parameters.

7.28 xd_read_zone_id

7.28.1 Overview

The **xd_read_zone_id** CFI function reads the list of zone names (Id) in a zone database file for Earth Explorer Missions.

7.28.2 Calling interface

The calling interface of the **xd_read_zone_id** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
       long status, num_zones;
       char *file_name;
       char **zone_ids
       long ierr[XD_NUM_ERR_READ_ZONE_ID];

       status = xd_read_zone_id (file_name,
                                 &num_zones, &zoned_ids,
                                 ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

       INTEGER*4 NUM_ZONES
       CHARACTER*LENGTH_NAME FILE_NAME, ZONE_IDS(NUM_ZONES)
       INTEGER*4 IERR(XD_NUM_ERR_READ_ZONE_ID), STATUS

       STATUS = XD_READ_ZONE_ID(FILE_NAME, NUM_ZONES, ZONE_IDS, IERR)
```

7.28.3 Input parameters

The **xd_read_zone_id** CFI function has the following input parameters:

Table 79: Input parameters of xd_read_zone_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*^	-	Zone database file name	-	-

7.28.4 Output parameters

The output parameters of the **xd_read_zone_id** CFI function are:

Table 80: Output parameters of xd_read_zone_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_zone_id	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
num_zones	long	-	Number of zones in the input file	-	-
zone_ids	char**	-	List fo zone names in the file	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *zone_ids* is a double pointer to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_zone_id**.

7.28.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_zone_id** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_zone_id** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 81: Error messages of xd_read_zone_id function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Zone File not found.	No calculation performed	XD_CFI_READ_ZONE_ID_INIT_PARSER_ERR	0
ERR	Data Block not found	No calculation performed	XD_CFI_READ_ZONE_ID_DATA_BLOCK_ERR	1
ERR	List_of_Zones not found.	No calculation performed	XD_CFI_READ_ZONE_ID_LIST_ZONES_READ_ERR	2

Table 81: Error messages of xd_read_zone_id function

Error type	Error message	Cause and impact	Error code	Error No
ERR	List_of_Zones attribute not read.	No calculation performed	XD_CFI_READ_ZONE_ID_LIST_ZONES_SIZE_ERR	3
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_ZONEI_D_MEMORY_ERR	4
ERR	Could not find the Zone_Id tag	No calculation performed	XD_CFI_READ_ZONE_ID_READ_ZONE_ERR	5

7.28.6 Runtime performances

The following runtime performances have been measured.

Table 82: Runtime performances of xd_read_zone_id function

Ultra Sparc II-400 [ms]
TBD

7.29 xd_free_zone_id

7.29.1 Overview

The **xd_free_zone_id** CFI function frees the memory allocated during the reading function **xd_read_zone_id**.

7.29.2 Calling interface

The calling interface of the **xd_free_zone_id** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      char** zone_ids;
      xd_free_zone_id (&zone_ids);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME ZONE_IDS(NUM_ZONES)
      XD_FREE_ZONE_ID (ZONE_IDS)
```

7.29.3 Input parameters

The **xd_free_zone_id** CFI function has the following input parameters:

Table 83: Input parameters of xd_free_zone_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
zone_ids	char**	-	Zone Id. list	-	-

7.29.4 Output parameters

This function does not return any value nor parameters.

7.30 xd_read_station

7.30.1 Overview

The **xd_read_station** CFI function reads the data of a station from a station database file.

7.30.2 Calling interface

The calling interface of the **xd_read_station** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
       long status;
       char *file_name, station_id;
       xd_station_rec station_rec;
       long ierr[XD_NUM_ERR_READ_STATION];

       status = xd_read_station (file_name, station_id,
                                 &station_rec, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

       CHARACTER*LENGTH_NAME FILE_NAME, STATION_ID
       XD_STATION_REC STATION_REC
       INTEGER*4 IERR(XD_NUM_ERR_READ_STATION), STATUS

       STATUS = XD_READ_STATION(FILE_NAME, STATION_ID,
&                               STATION_REC, IERR)
```

7.30.3 Input parameters

The **xd_read_station** CFI function has the following input parameters:

Table 84: Input parameters of xd_read_station function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Station database file name	-	-
station_id	char*	-	Station name (Id)	-	-

7.30.4 Output parameters

The output parameters of the **xd_read_station** CFI function are:

Table 85: Output parameters of xd_read_station function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_station	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
station_rec	xd_station_rec	-	Station record data	-	-
ierr	long[]	-	Error vector	-	-

7.30.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_station** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_station** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 86: Error messages of xd_read_station function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Ground Station DB File not found.	No calculation performed	XD_CFI_READ_STATION_INIT_PARSER_ERR	0
ERR	Data Block not found.	No calculation performed	XD_CFI_READ_STATION_DATA_BLOCK_ERR	1
ERR	Data Block attribute not read.	No calculation performed	XD_CFI_READ_STATION_DATA_BLOCK_ATTRIBUTE_ERR	2
ERR	Data Block not of XML type.	No calculation performed	XD_CFI_READ_STATION_XML_TYPE_ERR	3
ERR	List_of_Ground_Stations not found	No calculation performed	XD_CFI_READ_STATION_LIST_GS_READ_ERR	4
ERR	Number of ground stations negative.	No calculation performed	XD_CFI_READ_STATION_LIST_GS_SIZE_ERR	5

Table 86: Error messages of xd_read_station function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Internal error returned.	No calculation performed	XD_CFI_READ_STATION_INTERNAL_1_ERR	6
ERR	Cannot read Station_Id.	No calculation performed	XD_CFI_READ_STATION_STATION_ID_READ_ERR	7
ERR	Station id not found.	No calculation performed	XD_CFI_READ_STATION_STATION_ID_NOT_FOUND_ERR	8
ERR	Error reading station record	No calculation performed	XD_CFI_READ_STATION_REC_READ_ERR	9

7.30.6 Runtime performances

The following runtime performances have been measured.

Table 87: Runtime performances of xd_read_station function

Ultra Sparc II-400 [ms]
TBD

7.31 xd_read_station_file

7.31.1 Overview

The **xd_read_station_file** CFI function reads a whole station file for Earth Explorer Missions.

7.31.2 Calling interface

The calling interface of the **xd_read_station_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
       long status;
       char *file_name;
       xd_station_file station_data;
       long ierr[XD_NUM_ERR_READ_];

       status = xd_read_station_file (file_name,
                                      &station_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

       CHARACTER*LENGTH_NAME FILE_NAME
       XD_STATION_FILE STATION_DATA
       INTEGER*4 IERR(XD_NUM_ERR_READ_STATION_FILE), STATUS

       STATUS = XD_READ_STATION_FILE(FILE_NAME, STATION_DATA, IERR)
```

7.31.3 Input parameters

The **xd_read_station_file** CFI function has the following input parameters:

Table 88: Input parameters of xd_read_station_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Station database file name	-	-

7.31.4 Output parameters

The output parameters of the **xd_read_station_file** CFI function are:

Table 89: Output parameters of xd_read_station_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_station_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
station_data	xd_station_file	-	Station file data structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *station_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_station_file**.

7.31.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_station_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_station_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 90: Error messages of xd_read_station_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Ground Station DB File not found.	No calculation performed	XD_CFI_READ_STATION_FILE_INIT_PARSER_ERR	0
ERR	Data Block not found.	No calculation performed	XD_CFI_READ_STATION_FILE_DATA_BLOCK_ERR	1
ERR	Data Block attribute not read.	No calculation performed	XD_CFI_READ_STATION_FILE_DATA_BLOCK_ATTRIBUTE_ERR	2
ERR	Data Block not of XML type.	No calculation performed	XD_CFI_READ_STATION_FILE_XML_TYPE_ERR	3

Table 90: Error messages of xd_read_station_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	List_of_Ground_Stations not found.	No calculation performed	XD_CFI_READ_STATION_FILE_LIST_GS_READ_ERR	4
ERR	Number of ground stations negative.	No calculation performed	XD_CFI_READ_STATION_FILE_LIST_GS_SIZE_ERR	5
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_STATION_FILE_MEM_ERR	6
ERR	Error reading station record number %d	No calculation performed	XD_CFI_READ_STATION_FILE_REC_READ_ERR	7

7.31.6 Runtime performances

The following runtime performances have been measured.

Table 91: Runtime performances of xd_read_station_file function

Ultra Sparc II-400 [ms]
TBD

7.32 xd_free_station_file

7.32.1 Overview

The **xd_free_station_file** CFI function frees the memory allocated during the reading function **xd_read_station_file**.

7.32.2 Calling interface

The calling interface of the **xd_free_station_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      xd_station_file station_data;
      xd_free_station_file (&station_data);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      XD_STATION_FILE STATION_DATA
      XD_FREE_STATION_FILE(STATION_DATA)
```

7.32.3 Input parameters

The **xd_free_station_file** CFI function has the following input parameters:

Table 92: Input parameters of xd_free_station_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
station_data	xd_station_file	-	Station file data structure	-	-

7.32.4 Output parameters

This function does not return any value nor parameters.

7.33 xd_read_station_id

7.33.1 Overview

The **xd_read_station_id** CFI function reads the list of station names (Id) contained in a station datbase file.

7.33.2 Calling interface

The calling interface of the **xd_read_station_id** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status, num_stations;
      char *file_name;
      char **station_list;
      long ierr[XD_NUM_ERR_READ_STATION_ID];

      status = xd_read_station_id (file_name, &num_stations,
                                   &station_list, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      INTEGER NUM_STATIONS
      CHARACTER*LENGTH_NAME STATION_LIST(NUM_STATIONS)
      INTEGER*4 IERR(XD_NUM_ERR_READ_STATION_ID), STATUS

      STATUS = XD_READ_STATION_ID(FILE_NAME, NUM_STATIONS,
&                                 STATION_LIST, IERR)
```

7.33.3 Input parameters

The **xd_read_station_id** CFI function has the following input parameters:

Table 93: Input parameters of xd_read_station_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Station database file name	-	-

7.33.4 Output parameters

The output parameters of the **xd_read_station_id** CFI function are:

Table 94: Output parameters of xd_read_station_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_station_id	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
num_stations	long	-	Number of stations	-	-
station_list	char**	.	Station list name	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *station_list* is a double pointer to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_station_id**.

7.33.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_station_id** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_station_id** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 95: Error messages of xd_read_station_id function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Ground Station DB File not found.	No calculation performed	XD_CFI_READ_STATION_ID_INIT_PARSER_ERR	0
ERR	Data Block not found.	No calculation performed	XD_CFI_READ_STATION_ID_DATA_BLOCK_ERR	1
ERR	List_of_Ground_Stations not found.	No calculation performed	XD_CFI_READ_STATION_ID_LIST_GS_READ_ERR	2
ERR	Number of ground stations negative.	No calculation performed	XD_CFI_READ_STATION_ID_LIST_GS_SIZE_ERR	3

Table 95: Error messages of xd_read_station_id function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_STATION_ID_MEM_ERR	4
ERR	Error reading station Id.	No calculation performed	XD_CFI_READ_STATION_ID_READ_ID_ERR	5

7.33.6 Runtime performances

The following runtime performances have been measured.

Table 96: Runtime performances of xd_read_station_id function

Ultra Sparc II-400 [ms]
TBD

7.34 xd_free_station_id

7.34.1 Overview

The **xd_free_station_id** CFI function frees the memory allocated during the reading function **xd_read_station_id**.

7.34.2 Calling interface

The calling interface of the **xd_free_station_id** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      char **station_ids;
      xd_free_station_id (&station_ids);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME STATION_IDS(NUM_STATIONS)
      XD_FREE_STATION_ID (STATION_IDS)
```

7.34.3 Input parameters

The **xd_free_station_id** CFI function has the following input parameters:

Table 97: Input parameters of xd_free_station_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
station_ids	char **	-	Station Id list	-	-

7.34.4 Output parameters

This function does not return any value nor parameters.

7.35 xd_read_star

7.35.1 Overview

The **xd_read_star** CFI function reads the data for a star from a star database file.

7.35.2 Calling interface

The calling interface of the **xd_read_star** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name, star_id;
      xd_star_rec star_data;
      long ierr[XD_NUM_ERR_READ_STAR];

      status = xd_read_star (file_name, star_id, &star_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME, STAR_ID
      XD_STAR_REC STAR_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_STAR), STATUS

      STATUS = XD_READ_STAR(FILE_NAME, STAR_ID, STAR_DATA, IERR)
```

7.35.3 Input parameters

The **xd_read_star** CFI function has the following input parameters:

Table 98: Input parameters of xd_read_star function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Star database file name	-	-
star_id	char*	-	Star name (Id) to be read	-	-

7.35.4 Output parameters

The output parameters of the **xd_read_star** CFI function are:

Table 99: Output parameters of xd_read_star function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_star	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
star_data	xd_star_rec	-	Star data structure	-	-
ierr	long[]	-	Error vector	-	-

7.35.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_star** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_star** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 100: Error messages of xd_read_star function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Star database file not found: %s	No calculation performed	XD_CFI_READ_STAR_FILE_NOT_FOUND_ERR	0
ERR	star id. %s not found in the star database file	No calculation performed	XD_CFI_READ_STAR_STAR_NOT_FOUND_ERR	1

7.35.6 Runtime performances

The following runtime performances have been measured.

Table 101: Runtime performances of xd_read_star function

Ultra Sparc II-400 [ms]
TBD

7.36 xd_read_star_file

7.36.1 Overview

The **xd_read_star_file** CFI function reads a star database file for Earth Explorer Missions.

7.36.2 Calling interface

The calling interface of the **xd_read_star_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_star_file star_data;
      long ierr[XD_NUM_ERR_READ_STAR_FILE];

      status = xd_read_star_file (file_name, &star_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 STATUS
      CHARACTER*LENGTH_NAME FILE_NAME
      XD_STAR_FILE STAR_DATA
      INTEGER*4 IERR(XD_NUM_ERR_READ_STAR_FILE), STATUS

      STATUS = XD_READ_STAR_FILE(FILE_NAME, STAR_DATA, IERR)
```

7.36.3 Input parameters

The **xd_read_star_file** CFI function has the following input parameters:

Table 102: Input parameters of xd_read_star_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*^	-	Star database file name (full path)	-	-

7.36.4 Output parameters

The output parameters of the **xd_read_star_file** CFI function are:

Table 103: Output parameters of xd_read_star_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_star_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
star_data	xd_star_file	-	Star file structure	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *star_data* structure contains pointers to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data structure is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_star_file**.

7.36.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_star_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_star_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 104: Error messages of xd_read_star_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open the Star database file: %s	No calculation performed	XD_CFI_READ_STAR_FILE_FILE_NOT_FOUND_ERR	0
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_STAR_FILE_MEMORY_ERR	1
ERR	No stars found in file	No calculation performed	XD_CFI_READ_STAR_FILE_NO_STARS_ERR	2

7.36.6 Runtime performances

The following runtime performances have been measured.

Table 105: Runtime performances of xd_read_star_file function

Ultra Sparc II-400 [ms]
TBD

7.37 xd_read_star_id

7.37.1 Overview

The **xd_read_star_id** CFI function reads the list of star names from star database files.

7.37.2 Calling interface

The calling interface of the **xd_read_star_id** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      char **star_list;
      long num_stars;
      long ierr[XD_NUM_ERR_READ_STAR_ID];

      status = xd_read_star_id (file_name, &num_stars,
                                &star_list, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      INTEGER*4 STATUS, NUM_STARS
      CHARACTER*LENGTH_NAME FILE_NAME
      CHARACTER*LENGTH_NAME STAR_LIST(MAX_NUM_STARS)
      INTEGER*4 IERR(XD_NUM_ERR_READ_STAR_FILE), STATUS

      STATUS = XD_READ_STAR_ID(FILE_NAME, NUM_STARS, STAR_LIST, IERR)
```

7.37.3 Input parameters

The **xd_read_star_id** CFI function has the following input parameters:

Table 106: Input parameters of xd_read_star_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	Star database file	-	-

7.37.4 Output parameters

The output parameters of the **xd_read_star_id** CFI function are:

Table 107: Output parameters of xd_read_star_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_read_star_id	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
num_stars	long	-	Number of stars in the file	-	> 0
star_list	char**	-	Array of star names	-	-
ierr	long[]	-	Error vector	-	-

Memory Management: The *star_list* is a double pointer to memory allocated dinamically. In order to avoid memory leaks, the user will have to free that memory when the data is not to be used any more. The memory can be freed by calling to the CFI function **xd_free_star_id**.

7.37.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_read_star_id** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_read_star_id** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 108: Error messages of xd_read_star_id function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open the Star data-base file: %s	No calculation performed	XD_CFI_READ_STAR_ID_FILE_NOT_FOUND_ERR	0
ERR	Error allocating memory	No calculation performed	XD_CFI_READ_STAR_ID_MEMORY_ERR	1
ERR	No stars found in file	No calculation performed	XD_CFI_READ_STAR_ID_NO_STARS_ERR	2

7.37.6 Runtime performances

The following runtime performances have been measured.

Table 109: Runtime performances of xd_read_star_id function

Ultra Sparc II-400 [ms]
TBD

7.38 xd_write_orbit_file

7.38.1 Overview

The **xd_write_orbit_file** CFI function writes an orbit file in XML format using the data structure provided by the user. The orbit file can be either:

- A Predicted orbit file
- A Restituted orbit file
- A DORIS Predicted file

7.38.2 Calling interface

The calling interface of the **xd_write_orbit_file** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_fhr fhr;
      xd_orbit_file *osv_data;
      long ierr[XD_NUM_ERR_WRITE_ORBIT_FILE];

      status = xd_write_orbit_file(file_name, &fhr, &osv_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_FHR FHR
      XD_ORBIT_FILE OSV_DATA
      INTEGER*4 IERR(XD_NUM_ERR_WRITE_), STATUS

      STATUS = XD_WRITE_ORBIT_FILE(FILE_NAME, FHR, OSV_DATA, IERR)
```

7.38.3 Input parameters

The **xd_write_orbit_file** CFI function has the following input parameters:

Table 110: Input parameters of xd_write_orbit_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	File name for the orbit file. If empty string (i.e, ""), then the file is written with the name in the fixed_header structure (fhr)	-	-
fhr	xd_fhr	-	Fixed header structure	-	-
xd_orbit_file	osv_data	-	Orbital state vectors data structure	-	-

7.38.4 Output parameters

The output parameters of the **xd_write_orbit_file** CFI function are:

Table 111: Output parameters of xd_write_orbit_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_write_orbit_file	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
ierr	long[]	-	Error vector	-	-

7.38.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_write_orbit_file** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_write_orbit_file** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 112: Error messages of xd_write_orbit_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Cannot create root element	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_CREATE_TREE_ERR	0

Table 112: Error messages of xd_write_orbit_file function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Cannot create in-memory XML tree	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_CREATE_ROOT_ERR	1
ERR	Cannot write the fixed header	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_WRITE_FHR_ERR	2
ERR	Cannot add XML node to tree: %s	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_CREATE_NODE_ERR	3
ERR	Cannot convert time from processing to external	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_GET_ASCII_TIME_ERR	
ERR	annot write XML file	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_WRITE_ERR	
ERR	Cannot go to the desired node	No calculation performed	XD_CFI_WRITE_ORBIT_FILE_GOTO_NODE_ERR	

7.38.6 Runtime performances

The following runtime performances have been measured.

Table 113: Runtime performances of xd_write_orbit_file function

Ultra Sparc II-400 [ms]
TBD

7.39 xd_write_osf

7.39.1 Overview

The **xd_write_osf** CFI function writes an Orbit Scenario file in XML format using the data provided by the user.

7.39.2 Calling interface

The calling interface of the **xd_write_osf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_fhr fhr;
      xd_osf_file osf_data;
      long ierr[XD_NUM_ERR_WRITE_OSF];

      status = xd_write_osf (file_name, &fhr, &osf_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_FHR FHR
      XD_OSF_FILE OSF_DATA
      INTEGER*4 IERR(XD_NUM_ERR_WRITE_OSF), STATUS

      STATUS = XD_WRITE_OSF(FILE_NAME, FHR, OSF_DATA, IERR)
```

7.39.3 Input parameters

The **xd_write_osf** CFI function has the following input parameters:

Table 114: Input parameters of xd_write_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	File name for the orbit scenario file. If empty string (i.e, “”), then the file is written with the name in the fixed_header structure (fhr)	-	-
fhr	xd_fhr	-	Fixed header structure	-	-
xd_osf_file	osf_data	-	Orbital changes data structure	-	-

7.39.4 Output parameters

The output parameters of the **xd_write_osf** CFI function are:

Table 115: Output parameters of xd_write_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_write_osf	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
ierr	long[]	-	Error vector	-	-

7.39.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_write_osf** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_write_osf** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 116: Error messages of xd_write_osf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Cannot create in-memory XML tree	No calculation performed	XD_CFI_WRITE_OSF_CREATE_TREE_ERR	0

Table 116: Error messages of xd_write_osf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Cannot write the fixed header	No calculation performed	XD_CFI_WRITE_OSF_WRITE_FHR_ERR	1
ERR	Cannot create root element	No calculation performed	XD_CFI_WRITE_OSF_CREATE_ROOT_ERR	2
ERR	Cannot add XML node to tree	No calculation performed	XD_CFI_WRITE_OSF_CREATE_NODE_ERR	3
ERR	Cannot set XML node value	No calculation performed	XD_CFI_WRITE_OSF_SET_NODE_VALUE_ERR	4
ERR	Cannot convert time from processing to external	No calculation performed	XD_CFI_WRITE_OSF_TIME_TO_EXTERNAL_ERR	5
ERR	Cannot write XML file	No calculation performed	XD_CFI_WRITE_OSF_WRITE_ERR	6

7.39.6 Runtime performances

The following runtime performances have been measured.

Table 117: Runtime performances of xd_write_osf function

Ultra Sparc II-400 [ms]
TBD

7.40 xd_write_doris

7.40.1 Overview

The **xd_write_doris** CFI function writes a DORIS NAVIGATOR Product file for CRYOSAT, using the data provided by the user.

7.40.2 Calling interface

The calling interface of the **xd_write_doris** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_doris_mph_sph fhr;
      xd_doris_file doris_data;
      long ierr[XD_NUM_ERR_WRITE_DORIS];

      status = xd_write_doris (file_name, &fhr, &doris_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_DORIS_MPH_SPH FHR
      XD_DORIS_FILE DORIS_DATA
      INTEGER*4 IERR(XD_NUM_ERR_WRITE_DORIS), STATUS

      STATUS = XD_WRITE_DORIS(FILE_NAME, FHR, DORIS_DATA, IERR)
```

7.40.3 Input parameters

The **xd_write_doris** CFI function has the following input parameters:

Table 118: Input parameters of xd_write_doris function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	DORIS file name	-	-
fhr	xd_doris_mph_sph	-	Main and Specific product headers	-	-

Table 118: Input parameters of xd_write_doris function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
doris_data	xd_doris_file	-	DORIS data structure	-	-

7.40.4 Output parameters

The output parameters of the **xd_write_doris** CFI function are:

Table 119: Output parameters of xd_write_doris function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_write_doris	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
ierr	long[]	-	Error vector	-	-

7.40.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_write_doris** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_write_doris** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 120: Error messages of xd_write_doris function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open the file %s for writing	No calculation performed	XD_CFI_WRITE_DORIS_OPEN_ERR	0
ERR	Error writing the fixed header	No calculation performed	XD_CFI_WRITE_DORIS_WRITE_FHR_ERR	1
ERR	Error writing the binary data	No calculation performed	XD_CFI_WRITE_DORIS_WRITE_BINARY_ERR	2

7.40.6 Runtime performances

The following runtime performances have been measured.

Table 121: Runtime performances of xd_write_doris function

Ultra Sparc II-400 [ms]
TBD

7.41 xd_write_stf

7.41.1 Overview

The **xd_write_stf** CFI function writes a swath template file XML format using the data provided by the user.

7.41.2 Calling interface

The calling interface of the **xd_write_stf** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_fhr fhr;
      xd_stf_file stf_data;
      long ierr[XD_NUM_ERR_WRITE_STF];

      status = xd_write_stf (file_name, &fhr, &stf_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_FHR FHR
      XD_STF_FILE STF_DATA
      INTEGER*4 IERR(XD_NUM_ERR_WRITE_STF), STATUS

      STATUS = XD_WRITE_STF(FILE_NAME, FHR, STF_DATA, IERR)
```

7.41.3 Input parameters

The **xd_write_stf** CFI function has the following input parameters:

Table 122: Input parameters of xd_write_stf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	File name for the swath template file. If empty string (i.e, “”), then the file is written with the name in the fixed_header structure (fhr)	-	-
fhr	xd_fhr	-	Fixed header structure	-	-
xd_stf_file	stf_data	-	STF data structure	-	-

7.41.4 Output parameters

The output parameters of the **xd_write_stf** CFI function are:

Table 123: Output parameters of xd_write_stf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_write_stf	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
ierr	long[]	-	Error vector	-	-

7.41.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_write_stf** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_write_stf** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 124: Error messages of xd_write_stf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input file name. The file cannot be created	No calculation performed	XD_CFI_WRITE_STF_WRONG_FILENAME_ERR	0
ERR	Cannot create XML tree.	No calculation performed	XD_CFI_WRITE_STF_CREATE_XML_ERR	1
ERR	Cannot create root node in the XML tree.	No calculation performed	XD_CFI_WRITE_STF_CREATE_ROOT_XML_ERR	2
ERR	Error writing fixed header.	No calculation performed	XD_CFI_WRITE_STF_XD_FHR_WRITE_ERR	3
ERR	Error while writing Swath Template File variable header.	No calculation performed	XD_CFI_WRITE_STF_XD_STF_VHR_WRITE_ERR	4
ERR	Cannot create the node %s	No calculation performed	XD_CFI_WRITE_STF_CREATE_NODE_ERR	5
ERR	Wrong swath_type	No calculation performed	XD_CFI_WRITE_STF_WRONG_SWATH_TYPE_ERR	6
ERR	Error while writing the swath record n.%d	No calculation performed	XD_CFI_WRITE_STF_WRITE_REC_ERR	7
ERR	Cannot write to disk the XML tree	No calculation performed	XD_CFI_WRITE_STF_WRITE_ERR	8
ERR	Could not find the input directory \"%s\". The current directory will be used instead	No calculation performed	XD_CFI_WRITE_STF_NO_DIR_WARN	9

7.41.6 Runtime performances

The following runtime performances have been measured.

Table 125: Runtime performances of xd_write_stf function

Ultra Sparc II-400 [ms]
TBD

7.42 xd_write_att

7.42.1 Overview

The **xd_write_att** CFI function writes an attitude generic file in XML format using the data provided by the user.

7.42.2 Calling interface

The calling interface of the **xd_write_att** CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status;
      char *file_name;
      xd_fhr fhr;
      xd_att_file att_data;
      long ierr[XD_NUM_ERR_WRITE_ATT];

      status = xd_write_att (file_name, &fhr, &att_data, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILE_NAME
      XD_FHR FHR
      XD_ATT_FILE ATT_DATA
      INTEGER*4 IERR(XD_NUM_ERR_WRITE_ATT), STATUS

      STATUS = XD_WRITE_ATT(FILE_NAME, FHR, ATT_DATA, IERR)
```

7.42.3 Input parameters

The **xd_write_att** CFI function has the following input parameters:

Table 126: Input parameters of xd_write_att function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
file_name	char*	-	File name for the attitude file. If empty string (i.e, “”), then the file is written with the name in the fixed_header structure (fhr)	-	-
fhr	xd_fhr	-	Fixed header structure	-	-
xd_att_file	att_data	-	Attitude data structure	-	-

7.42.4 Output parameters

The output parameters of the **xd_write_att** CFI function are:

Table 127: Output parameters of xd_write_att function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_write_att	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
ierr	long[]	-	Error vector	-	-

7.42.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xd_write_att** CFI function after translating the returned error vector into the equivalent list of error messages by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_msg** (see [GEN_SUM]).

This table also indicates the type of message returned, i.e. either a warning (WARN) or an error (ERR), the cause of such a message and the impact on the performed calculation.

The table is completed by the error code and value. These error codes can be obtained translating the error vector returned by the **xd_write_att** function by calling the function of the EXPLORER_DATA_HANDLING software library **xd_get_code** (see [GEN_SUM])

Table 128: Error messages of xd_write_att function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Cannot create in-memory XML tree	No calculation performed	XD_CFI_WRITE_ATT_CREATE_TREE_ERR	0

Table 128: Error messages of xd_write_att function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Cannot create root element	No calculation performed	XD_CFI_WRITE_ATT_CREATE_ROOT_ERR	1
ERR	Cannot write the fixed header	No calculation performed	XD_CFI_WRITE_ATT_WRITE_FHR_ERR	2
ERR	Cannot add XML node to tree: %s	No calculation performed	XD_CFI_WRITE_ATT_CREATE_NODE_ERR	3
ERR	Cannot convert time from processing to external	No calculation performed	XD_CFI_WRITE_ATT_GET_ASCII_TIME_ERR	4
ERR	Cannot go to the desired node	No calculation performed	XD_CFI_WRITE_ATT_GOTO_NODE_ERR	5
ERR	Cannot write XML file	No calculation performed	XD_CFI_WRITE_ATT_WRITE_ERR	6

7.42.6 Runtime performances

The following runtime performances have been measured.

Table 129: Runtime performances of xd_write_att function

Ultra Sparc II-400 [ms]
TBD

7.43 xd_xml_validate

7.43.1 Overview

The **xd_xml_validate** CFI function validates an XML file using its XML schema and checks the XML schema versioning.

7.43.2 Calling interface

The calling interface of the CFI function is the following (input parameters are underlined):

```
#include <explorer_data_handling.h>
{
      long status, valid_status;
      char *filename, *schema, *logfile;
      long mode;
      long ierr[XD_NUM_ERR_XML_VALIDATE];

      status = xd_xml_validate (filename, &mode, &schema, &logfile,
                                valid_status, ierr);
}
```

For ForTran programs the declaration and calling procedure is as follows (input parameters are underlined, note that the C preprocessor must be used because of the presence of the `#include` statement):

```
#include <explorer_data_handling.inc>

      CHARACTER*LENGTH_NAME FILENAME, SCHEMA, LOGFILE
      INTEGER*4 VALID_STATUS, MODE
      INTEGER*4 IERR(XD_NUM_ERR_XML_VALIDATE), STATUS

      STATUS = XD_XML_VALIDATE(FILENAME, MODE, SCHEMA, LOGFILE,
                               VALID_STATUS, IERR)
```

7.43.3 Input parameters

The **xd_xml_validate** CFI function has the following input parameters:

Table 130: Input parameters of xd_xml_validate function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
filename	char*	-	File name to validate	-	-

Table 130: Input parameters of xd_xml_validate function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
mode	long	-	Flag to select the schema to be used to validate the file. It can be either: • XD_DEFAULT_SCHEMA: use the schema that is in the root element of the XML file. or • XD_USER_SCHEMA: use the schema given in the *schema* parameter in the interface.	-	-
schema	char*	-	Schema file	-	-
logfile	char*	-	Log file. It is used to store the messages returned by the validation process.	-	-

7.43.4 Output parameters

The output parameters of the **xd_xml_validate** CFI function are:

Table 131: Output parameters of xd_xml_validate function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xd_xml_validate	long	-	Function status flag: • = 0 No error • > 0 Warnings, results generated • < 0 Error, no results generated	-	-
valid_status	long	-	The result of the validation: • XD_XML_INVALID • XD_XML_VALID	-	-
ierr	long[]	-	Error vector	-	-

7.43.5 Warnings and errors

TBD

7.43.6 Runtime performances

The following runtime performances have been measured.

Table 132: Runtime performances of xd_xml_validate function

Ultra Sparc II-400 [ms]
TBD

7.43.7 Executable program

An XML file can also be validated using the executable program **xml_validate.** It can be called from a Unix shell as:

```
xml_validate -file filename
             [-sch schema_filename]
             [-log log_filename]
             [ -help ]
             [ -v ]
             [ -show ]
```

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- [-v] option for Verbose mode (default is Silent).
- [-show] displays the inputs of the function and the results.
- The filename is validated using the schema_filename if it is provided. If not, the default schema is used (the one in the root element of the file).
- The validation log is stored in the log_filename. By default the standard output is used.

8 FILES FORMAT SPECIFICATION

This section presents the formats for all the files used by the Earth Explorer CFI software.

The files used by the CFI can be:

- External: Files generated and/or used for the CFI software and other external facilities.
- Internal: Files used only in the CFI for configuration purposes.

All internal files are written in ASCII, with XML syntax. Following the usual format for the Earth Explorer Files, the file contains both:

- A header: It is divided in a fixed header and optionally a variable header. The format for the fixed header is common to all Earth Explorer Files and can be seen in [EE_FMT].
- A data block containing the input/output data of the functions.

The general structrure for a file will be:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
      ...
      </Fixed_Header>
      <Variable_Header>
      ...
      </Variable_Header>
   </Earth_Explorer_Header>

   <Data_Block type="xml">
   ...
   </Data_Block type>

</Earth_Explorer_File>
```

8.1 Fixed Header

8.1.1 Format

The Fixed Header is an XML structure. Many of its fields are redundant with the File Name elements, but are present in more readable form in the Fixed Header, whereas in File Name they are more compact for obvious reasons. Its format is described in the followig tables:

Table 133: Fixed Header Structure

Tag name	type	Attribute	C Format	Description
File_Name	string	-	%s	It is a repetition of the Logical File Name, i.e. the File Names excluding the extension. This allows this field to be independent from the storage in 1 complete file or 2 separate files for Header and Data Block
File_Description	string	-	%s	A 1-line description of the File Type. Each Mission shall define the list of official file descriptions (per File Type).

Table 133: Fixed Header Structure

Tag name	type	Attribute	C Format	Description
Notes	string	-	%s	Multi-lines free text. This can be used for any type of comment, relevant that instance of the file.
Mission	string	-	%s	A 1-word description of the Mission, coherent with the Mission element in the File Name. See [EE_FMT] Section 4.1.1 for the official list.
File_Class	string	-	%10s	A 1-line description of the file class, coherent with the File Class element in the File Name. Each Mission shall define the list of official file classes.
File_Type	string	-	%s	It is a repetition of the File Type element in the File Name.
Validity_Period	structure (see table 134)	-	-	Structure containing the start-stop validity period of the file.
File_Version	integer	-	%04ld	It is a repetition of the File Version element in the File Name. Must start at 1 (not 0).
Source	structure (see table 135)	-	-	Structure with information about the source of the file.

Table 134: Fixed Header. Validity Period

Tag name	type	Attribute	C Format	Description
Validity_Start	%23s	-	string	This is the UTC Validity Start Time, coherent with the Validity Start Time in the File Name, but in CCSDS ASCII format with time reference. Note that this can have the special value indicating “beginning of mission” (without an absolute time specified) as defined in [MCD]
Validity_Stop	%23s	-	string	This is the UTC Validity Stop Time, coherent with the Validity Stop Time in the File Name, but in CCSDS ASCII format with time reference. Note that this can have the special value indicating “end of mission” (without an absolute time specified) as defined in [MCD].

Table 135: Fixed Header. Source

Tag name	type	Attribute	C Format	Description
System	string	-	%s	Name of the Ground Segment element creating the file (e.g. FOS, PDS, SSALTO...)
Creator	string	-	%s	Name of the tool, within the Ground Segment element, creating the file (e.g. CS-MCS, IPF1...)
Creator_Version	string	-	%s	Version of the tool (e.g. 1.0, 2.1a ...)
Creation_Date	string	-	%23s	This is the UTC Creation Date, in CCSDS ASCII format with time reference. This format is defined in [MCD].

8.1.2 Example

```
<Fixed_Header>
        <File_Name>logical file name</File_Name>
        <File_Description>1-line file description</File_Description>
        <Notes>
              free text, free format
              several lines if needed
        </Notes>
        <Mission>mission name</Mission> (e.g. Cryosat)
        <File_Class>1-line file class description</File_Class>
        <File_Type>TTTTTTTTTT</File_Type>
        <Validity_Period>
              <Validity_Start>UTC=yyyy-mm-ddThh:mm:ss</Validity_Start>
              <Validity_Stop>UTC=yyyy-mm-ddThh:mm:ss</Validity_Stop>
        </Validity_Period>
        <File_Version>vvvv</File_Version>
        <Source>
              <System>name of system creating the file</System>
              <Creator>name of tool creating the file</Creator>
              <Creator_Version>version of tool</Creator_Version>
              <Creation_Date>UTC=yyyy-mm-ddThh:mm:ss</Creation_Date>
        </Source>
</Fixed_Header>
```

8.2 Predicted Orbit files

8.2.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 136: Predicted Orbit File. Data_Block

Tag name	type	Attribute	C Format	Description
List_of_OSVs	List of <OSV> Structures (See table 137)	count="*n*" where *n* is the number of elements in the list	-	List of Orbit State Vectors

Table 137: Precicted Orbit File. OSV

Tag name	type	Attribute	C Format	Description
TAI	date		%s	TAI date and time of OSV, in ASCII standard time format, including time reference and micro-seconds
UTC	date		%s	UTC date and time of OSV, in ASCII standard time format, including time reference and micro-seconds
UT1	date		%s	UT1 date and time of OSV, in ASCII standard time format, including time reference and micro-seconds
Absolute_Orbit	integer		%+05ld	absolute orbit counter This counter is incremented by one unit from a record to the next. It must be differenciated with the real absolute orbit number on which the state vector really belongs i.e : if the Z value of the OSV is >= 0 then "real" absolute orbit number equal the absolute orbit counter if the Z value of the OSV is < 0 then "real" absolute orbit number equal the absolute orbit counter minus 1.
X	real	m	%+012.3lf	X position in earth-fixed coordinate system
Y	real	m	%+012.3lf	Y position in earth-fixed coordinate system
Z	real	m	%+012.3lf	Z position in earth-fixed coordinate system
VX	real	m/s	%+012.6lf	X velocity in earth-fixed coordinate system
VY	real	m/s	%+012.6lf	Y velocity in earth-fixed coordinate system
VZ	real	m/s	%+012.6lf	Z velocity in earth-fixed coordinate system

Table 137: Precicted Orbit File. OSV

Tag name	type	Attribute	C Format	Description
Quality	string	string	%s13	Values is/are TBD. This parameter is added to keep format compatibiolity withe DORIS Precise Orbit File Format. Default (“not used”) value is “0000000000000”

8.2.2 Example

```
<?xml version ="1.0"?>
<Earth_Explorer_File>
        <Earth_Explorer_Header>
              <Fixed_Header>
              <File_Name>CS_OPER_MPL_ORBPRE_20020315T205400_20020321T205500_0001</File_Name>
              <File_Description>FOS Predicted Orbit File</File_Description>
              <Notes></Notes>
              <Mission>CryoSat</Mission>
              <File_Class>Routine Operations</File_Class>
              <File_Type>MPL_ORBPRE</File_Type>
              <Validity_Period>
                    <Validity_Start>UTC=2002-03-15T20:54:44</Validity_Start>
                    <Validity_Stop>UTC=2002-03-21T20:54:44</Validity_Stop>
              </Validity_Period>
              <File_Version>0001</File_Version>
              <Source>
                    <System>FOS</System>
                    <Creator>name of tool creating the file</Creator>
                    <Creator_Version>1.0</Creator_Version>
                    <Creation_Date>UTC=2002-03-14T14:00:00</Creation_Date>
              </Source>
              </Fixed_Header>
              <Variable_Header>
              </Variable_Header>
        </Earth_Explorer_Header>
        <Data_Block type=”xml”>
              <List_of_OSVs count=”n”>
                    <OSV>
                          <TAI>TAI=2002-03-15T20:54:44.069916</TAI>
                          <UTC>UTC=2002-03-15T20:54:04.069916</UTC>
                          <UT1>UT1=2002-03-15T20:54:04.049916</UT1>
                          <Absolute_Orbit>+00212</Absolute_Orbit>
                          <X unit=”m”>+6874869.308</X>
                          <Y unit=”m”>+2033241.443</Y>
                          <Z unit=”m”>-0000995.334</Z>
                          <VX unit=”m/s”>+0453.224305</VX>
                          <VY unit=”m/s”>-1567.965124</VY>
                          <VZ unit=”m/s”>+7374.880929</VZ>
                          <Quality>0000000000000</Quality>
                    </OSV>
                    <OSV>
                          <TAI>TAI=2002-03-15T22:35:24.246686</TAI>
                          <UTC>UTC=2002-03-15T22:34:44.246686</UTC>
                          <UT1>UT1=2002-03-15T22:34:44.046686</UT1>
                          <Absolute_Orbit>+00213</Absolute_Orbit>
                          <X unit=”m”>+7086938.577</X>
```

```
                        <Y unit="m">-1083333.239</Y>
                        <Z unit="m">-0001004.069</Z>
                        <VX unit="m/s">-0256.608063</VX>
                        <VY unit="m/s">-1611.943172</VY>
                        <VZ unit="m/s">+7374.846086</VZ>
                        <Quality>0000000000000</Quality>
                    </OSV>
                    ...
                </List_of_OSVs>
            </Data_Block>
</Earth_Explorer_File>
```

8.3 Restituted Orbit files

Format is equal to the Predicted Orbit File, see section 8.2.

8.4 Doris Preliminary/Precisefiles

Format is equal to the Predicted Orbit File, see section 8.2.

8.5 Orbit Scenario files

8.5.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 138: Orbit Scenario File. Data_Block

Tag name	type	Attribute	C Format	Description
List_of_Orbit_Changes	List of <Orbit_Change> Structures (See table 139)	count="*n*" where *n* is the number of elements in the list	-	List of Orbital Changes

Table 139: Orbit Scenario File. Orbit_Change

Tag name	type	Attribute	C Format	Description
Orbit	structure (see table 140)			Orbit information
Cycle	structure (see table 141)			Cycle information
Time_of_ANX	structure (see table 142)			Ascending node time

Table 140: Orbit Scenario File. Orbit

Tag name	type	Attribute	C Format	Description
Absolute_Orbit	integer		%ld	absolute orbit counter.

Table 140: Orbit Scenario File. Orbit

Tag name	type	Attribute	C Format	Description
Relative_Orbit	integer		%ld	relative orbit number
Cycle_Number	integer		%ld	cycle number; incremented after each new repeat cycle
Phase_Number	integer		%ld	phase number; incremented on Mission Management decision

Table 141: Orbit Scenario File. Cycle

Tag name	type	Attribute	C Format	Description
Repeat_Cycle	integer		%ld	
Cycle_Length	integer		%ld	
ANX_Longitude	real	deg	%+011.6lf	longitude of ascending node crossing (ANX)
MLST	date		%s	mean local solar time at ANX of relative orbit 1
MLST_Drift	real	s/day	%+12.6lf	drift of mean local solar time over 1 orbit

Table 142: Orbit Scenario File. Time_of_ANX

Tag name	type	Attribute	C Format	Description
TAI	date		%s	TAI date and time of ANX, in ASCII CCSDStime format, including time reference and micro-seconds
UTC	date		%s	UTC date and time of ANX, in ASCII CCSDS time format, including time reference and micro-seconds
UT1	date		%s	UT1 date and time of ANX, in ASCII CCSDS time format, including time reference and micro-seconds

8.5.2 Example

```
<?xml version ="1.0"?>
<Earth_Explorer_File>
        <Earth_Explorer_Header>
             <Fixed_Header>
             <File_Name>CS_OPER_MPL_ORBSCT_20020312T140002_99999999T999999_0001</File_Name>
             <File_Description>Orbit Scenario File</File_Description>
             <Notes></Notes>
             <Mission>CryoSat</Mission>
             <File_Class>Routine Operations</File_Class>
             <File_Type>MPL_ORBSCT</File_Type>
             <Validity_Period>
                  <Validity_Start>UTC=2002-03-12T14:00:02</Validity_Start>
                  <Validity_Stop>UTC=9999-99-99T99:99:99</Validity_Stop>
             </Validity_Period>
             <File_Version>0001</File_Version>
```

```
            <Source>
                  <System>RPF</System>
                  <Creator>name of tool creating the file</Creator>
                  <Creator_Version>1.0</Creator_Version>
                  <Creation_Date>UTC=2002-03-10T14:00:00</Creation_Date>
            </Source>
            </Fixed_Header>
            <Variable_Header>
            </Variable_Header>
      </Earth_Explorer_Header>
      <Data_Block type="xml">
            <List_of_Orbit_Changes count="2">
                  <Orbit_Change>
                        <Orbit>
                              <Absolute_Orbit>1</Absolute_Orbit>
                              <Relative_Orbit>1</Relative_Orbit>
                              <Cycle_Number>1</Cycle_Number>
                              <Phase_Number>1</Phase_Number>
                        </Orbit>
                        <Cycle>
                              <Repeat_Cycle unit="day">369</Repeat_Cycle>
                              <Cycle_Length unit="orbit">5344</Cycle_Length>
                              <ANX_Longitude unit="deg">+023.600000</ANX_Longitude>
                              <MLST>22:17:19.999999</MLST>
                              <MLST_Drift unit="s/day">+9.000000</MLST_Drift>
                        </Cycle>
                        <Time_of_ANX>
                              <TAI>TAI=2001-03-12T14:00:34.999999</TAI>
                              <UTC>UTC=2001-03-12T14:00:02.999999</UTC>
                              <UT1>UT1=2001-03-12T14:00:02.777777</UT1>
                        </Time_of_ANX>
                  </Orbit_Change>
                  <Orbit_Change>
                        <Orbit>
                              <Absolute_Orbit>1</Absolute_Orbit>
                              <Relative_Orbit>1</Relative_Orbit>
                              <Cycle_Number>10</Cycle_Number>
                              <Phase_Number>1</Phase_Number>
                        </Orbit>
                        <Cycle>
                              <Repeat_Cycle unit="day">2</Repeat_Cycle>
                              <Cycle_Length unit="orbit">29</Cycle_Length>
                              <ANX_Longitude unit="deg">+023.600000</ANX_Longitude>
                              <MLST>22:17:19.999999</MLST>
                              <MLST_Drift unit="s/day">+9.000000</MLST_Drift>
                        </Cycle>
                        <Time_of_ANX>
                              <TAI>TAI=2001-04-01T14:00:34.999999</TAI>
                              <UTC>UTC=2001-04-01T14:00:02.999999</UTC>
                              <UT1>UT1=2001-04-01T14:00:02.777777</UT1>
                        </Time_of_ANX>
                  </Orbit_Change>
            </List_of_Orbit_Changes>
      </Data_Block>
</Earth_Explorer_File>
```

8.6 DORIS Navigator files

A DORIS Navigator file consist in two files, the header file and the data block file. They are compliant with [PDS_FMT]

8.7 Star Tracker files

A Star tracker file consists in a couple of files: the CryoSat standard header file and the data block file. They are compliant with [PDS_FMT]

8.8 Satellite Configuration File

8.8.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 143: Satellite Configuration File. Data Block

Tag name	type	Attribute	C Format	Description
Satellite_Name	string	-	%s	Satellite Name
Lib_Init	structure (see table 144)	-	-	Low and tight tolerances for orbital parameters
Orbit_Init	structure (see table 145)	-	-	Default Orbital parameters

Table 144: Satellite Configuration File. Lib_Init Structure

Tag name	type	Attribute	C Format	Description
Low_Tolerances	structure (see table 146)	-	-	Low tolerances for orbital parameters
Tight_Tolerances	structure (see table 143)	-	-	Tight tolerances for orbital parameters

Table 145: Satellite Configuration File. Orbit_InitStructure

Tag name	type	Attribute	C Format	Description
Min_Semi_Major_Axis	real	-	%lf	Minimum semi-major axis (meters)
Nom_Semi_Major_Axis	real	-	%lf	Nominal semi-major axis (meters)
Max_Semi_Major_Axis	real	-	%lf	Maximum semi-major axis (meters)
Min_Inclination	real	-	%lf	Minimum inclination (degrees)
Nom_Inclination	real	-	%lf	Nominal inclination (degrees)
Max_Inclination	real	-	%lf	Maximum inclination (degrees)
Nom_Eccentricity	real	-	%lf	Nominal Eccentricity
Nom_Arg_Perigee	real	-	%lf	Nominal Argument of perigee (degrees)

Table 146: Satellite Configuration File. Low and Tight Tolerances Structure

Tag name	type	Attribute	C Format	Description
Min_Semi_Major_Axis	real	-	%lf	Minimum semi-major axis (meters)
Max_Semi_Major_Axis	real	-	%lf	Maximum semi-major axis (meters)
Min_Inclination	real	-	%lf	Minimum inclination (degrees)

Table 146: Satellite Configuration File. Low and Tight Tolerances Structure

Tag name	type	Attribute	C Format	Description
Max_Inclination	real	-	%lf	Maximum inclination (degrees)
Min_Eccentricity	real	-	%lf	Eccentricity
Max_Eccentricity	real	-	%lf	Eccentricity

8.8.2 File Example

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
        <File_Name>Cryosat_configuration_file.xml</File_Name>
        <File_Description>Satellite Configurantion File</File_Description>
        <Notes/>
        <Mission>Cryosat</Mission>
        <File_Class>TEST</File_Class>
        <File_Type></File_Type>
        <Validity_Period>
          <Validity_Start>UTC=0000-00-00T00:00:00.000000</Validity_Start>
          <Validity_Stop>UTC=9999-99-99T99:99:99.5999999</Validity_Stop>
      </Validity_Period>
      <File_Version>1</File_Version>
      <Source>
        <System></System>
        <Creator></Creator>
        <Creator_Version></Creator_Version>
        <Creation_Date>UTC=2003-11-28T17:25:44</Creation_Date>
      </Source>
      </Fixed_Header>
      <Variable_Header\>
   </Earth_Explorer_Header>
   <Data_Block type="xml">
         <Satellite_Name>CryoSat</Satellite_Name>
         <Lib_init>
               <Low_Tolerances>
                     <Min_Semi_Major_Axis>1000000.0</Min_Semi_Major_Axis>
                     <Max_Semi_Major_Axis>10000000.0</Max_Semi_Major_Axis>
                     <Min_Inclination>60.0</Min_Inclination>
                     <Max_Inclination>120.0</Max_Inclination>
                     <Min_Eccentricity>0.0</Min_Eccentricity>
                     <Max_Eccentricity>0.5</Max_Eccentricity>
               </Low_Tolerances>
               <Tight_Tolerances>
                     <Min_Semi_Major_Axis>1000000.0</Min_Semi_Major_Axis>
                     <Max_Semi_Major_Axis>10000000.0</Max_Semi_Major_Axis>
                     <Min_Inclination>60.0000</Min_Inclination>
                     <Max_Inclination>120.0000</Max_Inclination>
                     <Min_Eccentricity>0.000</Min_Eccentricity>
                     <Max_Eccentricity>0.500</Max_Eccentricity>
               </Tight_Tolerances>
         </Lib_init>
         <Orbit_init>
               <Min_Semi_Major_Axis>7055200.0</Min_Semi_Major_Axis>
               <Nom_Semi_Major_Axis>7096643.0</Nom_Semi_Major_Axis>
               <Max_Semi_Major_Axis>7131206.0</Max_Semi_Major_Axis>
```

```
            <Min_Inclination>91.8981</Min_Inclination>
            <Nom_Inclination>92.0000</Nom_Inclination>
            <Max_Inclination>92.0732</Max_Inclination>
            <Nom_Eccentricity>0.0013</Nom_Eccentricity>
            <Nom_Arg_Perigee>90.0</Nom_Arg_Perigee>
        </Orbit_init>
</Data_Block>
</Earth_Explorer_File>
```

8.9 Attitude File

8.9.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 147: Attitude File. Data Block

Tag name	type	Attribute	C Format	Description
Attitude_File_Type	string	-	%s	The initial attitude frame. It can be: • Sat_Nominal_Attitude • Sat_Attitude • Instr_Attitude
Attitude_Data_Type	string	-	%s	It defines the type of attitude data: • Quaternions • Attitude_Angles
Max_Gap		unit="s"		Maximum gap between two consecutive set of angles or quaternions
Attitude_Angles_Data or Quaternions_Data	Structures: see table 148 for the angles data or table 149 for the quaternions	-	-	Structure for the list of angles or the quaternions

Table 148: Attitude File. Attitude Angles Data

Tag name	type	Attribute	C Format	Description
List_of_Attitude_Angles	List of Attitude_Angles (see table 150)	count="*n*"	-	List of Attitude_Angles

Table 149: Attitude File. Quaternions Data

Tag name	type	Attribute	C Format	Description
List_of_Quaternions	List of Quaternions (see table 151)	count="*n*"	-	List of Quaternions

Table 150: Attitude File. List of Attitude Angles

Tag name	type	Attribute	C Format	Description
Attitude_Angles	Structure (see table 152)	-	-	Pitch, roll and yaw angles for a given time

Table 151: Attitude File. List of Quaternions Data

Tag name	type	Attribute	C Format	Description
Quaternions	Structure (see table 153)	-	-	Set of quaternions for a given time

Table 152: Attitude File. Attitude_Angles

Tag name	type	Attribute	C Format	Description
Time	string	ref="*RRR*" where RRR stands for: • TAI • UTC • UT1 • GPS	%s	Date for the angles. The date format is CCSDS-A with reference and microseconds (RRR=yyyy-mm-ddThh:nn:ss.uuuuuu)
Pitch	real	unit="deg"	%lf	Pitch angle
Roll	real	unit="deg"	%lf	Roll angle
Yaw	real	unit="deg"	%lf	Yaw angle

Table 153: Attitude File. Quaternions

Tag name	type	Attribute	C Format	Description
Time	string	ref="*RRR*" where RRR stands for: • TAI • UTC • UT1 • GPS	%s	Date for the angles. The date format is CCSDS-A with reference and microseconds (RRR=yyyy-mm-ddThh:nn:ss.uuuuuu)
Q1	real	-	%lf	Quaternion
Q2	real	-	%lf	Quaternion
Q3	real	-	%lf	Quaternion
Q4	real	-	%lf	Quaternion

8.9.2 File Example

```
<?xml version="1.0"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
         <File_Name>ATT_TEST_FILE</File_Name>
         <File_Description>Attitude File</File_Description>
         <Notes/>
         <Mission>XXXXX</Mission>
         <File_Class>TEST</File_Class>
         <File_Type></File_Type>
         <Validity_Period>
            <Validity_Start>UTC=2002-03-03T08:09:17.232850</Validity_Start>
            <Validity_Stop>UTC=2002-03-03T09:48:23.505544</Validity_Stop>
         </Validity_Period>
         <File_Version>0101</File_Version>
```

```
        <Source>
           <System>CFI Acceptance</System>
           <Creator></Creator>
           <Creator_Version></Creator_Version>
           <Creation_Date>UTC=2003-11-28T17:25:44</Creation_Date>
        </Source>
     </Fixed_Header>
     <Variable_Header/>
  </Earth_Explorer_Header>
  <Data_Block type="xml">
     <Attitude_File_Type>Sat_Attitude</Attitude_File_Type>
     <Attitude_Data_Type>Attitude_Angles</Attitude_Data_Type>
     <Max_Gap unit="s">200</Max_Gap>
     <Attitude_Angles_Data>
        <List_of_Attitude_Angles count="5">
           <Attitude_Angles>
              <Time ref="TAI">TAI=2004-07-04T18:26:30.000000</Time>
              <Pitch unit="deg">0.05</Pitch>
              <Roll unit="deg">0.15</Roll>
              <Yaw unit="deg">-0.25</Yaw>
           </Attitude_Angles>
           <Attitude_Angles>
              <Time ref="TAI">TAI=2004-07-04T18:26:32.000000</Time>
              <Pitch unit="deg">0.07</Pitch>
              <Roll unit="deg">0.17</Roll>
              <Yaw unit="deg">-0.27</Yaw>
           </Attitude_Angles>
           <Attitude_Angles>
              <Time ref="TAI">TAI=2004-07-04T18:26:34.000000</Time>
              <Pitch unit="deg">0.09</Pitch>
              <Roll unit="deg">0.19</Roll>
              <Yaw unit="deg">-0.29</Yaw>
           </Attitude_Angles>
           <Attitude_Angles>
              <Time ref="TAI">TAI=2004-07-04T18:26:36.000000</Time>
              <Pitch unit="deg">0.11</Pitch>
              <Roll unit="deg">0.21</Roll>
              <Yaw unit="deg">-0.31</Yaw>
           </Attitude_Angles>
           <Attitude_Angles>
              <Time ref="TAI">TAI=2004-07-04T18:26:38.000000</Time>
              <Pitch unit="deg">0.13</Pitch>
              <Roll unit="deg">0.23</Roll>
              <Yaw unit="deg">-0.33</Yaw>
           </Attitude_Angles>
           <Attitude_Angles>
              <Time ref="TAI">TAI=2004-07-04T18:26:40.000000</Time>
              <Pitch unit="deg">0.15</Pitch>
              <Roll unit="deg">0.25</Roll>
              <Yaw unit="deg">-0.35</Yaw>
           </Attitude_Angles>
        </List_of_Attitude_Angles>
     </Attitude_Angles_Data>
  </Data_Block>
</Earth_Explorer_File>
```

8.10 Star tracker configuration File

8.10.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below. As it is a quite long file, only the relevant part to the CFIs are described.

Table 154: Star Tracker Configuration File. Data Block

Tag name	type	Attribute	C Format	Description
Satellite_Name	string	-	%s	Satellite Name
Mispointing	Structure (See table 155)	-	-	Set of rotation angles needed for mispointing computation

Table 155: Star Tracker Configuration File. Mispointing

Tag name	type	Attribute	C Format	Description
Aberration_Correction	string	-	%s	Aberration correction flag. Possible values are: • Yes: for applying the aberration-correction. • No: for not applying the aberration correction. • Reverse: for applying the aberration correction with the transposed matrix.
Star_Trackers_Limits	Structure (See table 156)	-	-	Limits for the validity fo the quaternions
Star_Trackers_Priority	Structure (See table 157)	-	-	Star trackers priority
List_of_Star_Trackers	Structure (See table 158)	count="*n*"	-	List of rotation angles from the antenna bench to the star trackers frame
Satellite_Mechanical_To_Antenna_Bench	Structure (See table 159)	-	-	Rotation angles from the satellite mechanical to the antenna bench frame
Satellite_Control_To_Satellite_Mechanical	Structure (See table 160)	-	-	Rotation angles from the satellite control to the satellite mechanical frame
Satellite_Attitude_To_Satellite_Control	Structure (See table 155)	-	-	Rotation angles from the satellite control to the satellite attitude frame

Table 156: Star Tracker Configuration File. Star tracker limits

Tag name	type	Attribute	C Format	Description
Max_Penalty	integer	-	%d	Maximum penalty for the quaternions

Table 156: Star Tracker Configuration File. Star tracker limits

Tag name	type	Attribute	C Format	Description
Quaternion_Norm_Threshold	real	-	%f	Threshold for the modulus of the quaternion
Max_Time_Gap	real	unit="s"	%f	Maximum time gap between two consequtive quaternions

Table 157: Star Tracker Configuration File. Star_Trackers_Priority

Tag name	type	Attribute	C Format	Description
File_Type_1	string	-	%s	
File_Type_2	string	-	%s	
File_Type_3	string	-	%s	

Table 158: Star Tracker Configuration File. List_of_Star_Trackers

Tag name	type	Attribute	C Format	Description
Star_Tracker	Structure (See table 159)	-	-	Antenna bench to Star tracker rotation angles

Table 159: Star Tracker Configuration File. Pre and Post Launch angles

Tag name	type	Attribute	C Format	Description
Pre_Launch_Angles	Structure (See table 160)	-	-	pre-launch angles
Post_Launch_Misalignment	Structure (See table 160)	-	-	post-launch angles

Table 160: Star Tracker Configuration File. Rotation_Angles

Tag name	type	Attribute	C Format	Description
X_Rotation	real	unit="deg"	%f	Rotation around the X-axis
Y_Rotation	real	unit="deg"	%f	Rotation around the Y-axis
Z_Rotation	real	unit="deg"	%f	Rotation around the Z-axis

8.10.2 File Example

```
<?xml version="1.0"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
   </Earth_Explorer_Header>
   <Data_Block type="xml">
         <Aberration_Correction>Yes</Aberration_Correction>
         <Satellite_Name>CryoSat</Satellite_Name>
         <Mispointing>
            <Star_Trackers_Limits>
               <Max_Penalty>5</Max_Penalty>
               <Quaternion_Norm_Threshold>0.000001</Quaternion_Norm_Threshold>
               <Max_Time_Gap unit="s">600</Max_Time_Gap>
            </Star_Trackers_Limits>
```

```
<Star_Trackers_Priority>
   <File_Type_1>STR1ATT_0_</File_Type_1>
   <File_Type_2>STR2ATT_0_</File_Type_2>
   <File_Type_3>STR3ATT_0_</File_Type_3>
</Star_Trackers_Priority>
<!-- Antenna Bench To Star Tracker rotation angles -->
<List_of_Star_Trackers count="3">
   <Star_Tracker>
      <Pre_Launch_Angles>
         <X_Rotation unit="deg">0.000</X_Rotation>
         <Y_Rotation unit="deg">0.000</Y_Rotation>
         <Z_Rotation unit="deg">0.000</Z_Rotation>
      </Pre_Launch_Angles>
      <Post_Launch_Misalignment>
         <X_Rotation unit="deg">0.000</X_Rotation>
         <Y_Rotation unit="deg">0.000</Y_Rotation>
         <Z_Rotation unit="deg">0.000</Z_Rotation>
      </Post_Launch_Misalignment>
   </Star_Tracker>
   <Star_Tracker>
      <Pre_Launch_Angles>
         <X_Rotation unit="deg">65.000</X_Rotation>
         <Y_Rotation unit="deg">0.000</Y_Rotation>
         <Z_Rotation unit="deg">0.000</Z_Rotation>
      </Pre_Launch_Angles>
      <Post_Launch_Misalignment>
         <X_Rotation unit="deg">0.000</X_Rotation>
         <Y_Rotation unit="deg">0.000</Y_Rotation>
         <Z_Rotation unit="deg">0.000</Z_Rotation>
      </Post_Launch_Misalignment>
   </Star_Tracker>
   <Star_Tracker>
      <Pre_Launch_Angles>
         <X_Rotation unit="deg">295.000</X_Rotation>
         <Y_Rotation unit="deg">0.000</Y_Rotation>
         <Z_Rotation unit="deg">0.000</Z_Rotation>
      </Pre_Launch_Angles>
      <Post_Launch_Misalignment>
         <X_Rotation unit="deg">0.000</X_Rotation>
         <Y_Rotation unit="deg">0.000</Y_Rotation>
         <Z_Rotation unit="deg">0.000</Z_Rotation>
      </Post_Launch_Misalignment>
   </Star_Tracker>
</List_of_Star_Trackers>
<!-- End Antenna Bench To Star Tracker rotation angles -->
<Satellite_Mechanical_To_Antenna_Bench>
<Pre_Launch_Angles>
   <X_Rotation unit="deg">0.000</X_Rotation>
   <Y_Rotation unit="deg">354.000</Y_Rotation>
   <Z_Rotation unit="deg">0.000</Z_Rotation>
</Pre_Launch_Angles>
<Post_Launch_Misalignment>
   <X_Rotation unit="deg">0.000</X_Rotation>
   <Y_Rotation unit="deg">0.000</Y_Rotation>
   <Z_Rotation unit="deg">0.000</Z_Rotation>
</Post_Launch_Misalignment>
</Satellite_Mechanical_To_Antenna_Bench>
<Satellite_Control_To_Satellite_Mechanical>
   <X_Rotation unit="deg">0.000</X_Rotation>
   <Y_Rotation unit="deg">6.000</Y_Rotation>
   <Z_Rotation unit="deg">0.000</Z_Rotation>
```

```
            </Satellite_Control_To_Satellite_Mechanical>
            <Satellite_Attitude_To_Satellite_Control>
               <X_Rotation unit="deg">0.000</X_Rotation>
               <Y_Rotation unit="deg">0.000</Y_Rotation>
               <Z_Rotation unit="deg">270.000</Z_Rotation>
            </Satellite_Attitude_To_Satellite_Control>
         </Mispointing>
         [...]
   </Data_Block>
</Earth_Explorer_File>
```

8.11 DEM Configuration File

8.11.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 161: DEM Configuration File. Data Block

Tag name	type	Attribute	C Format	Description
The following options are valid: • ACE_Model	Structure (see table 162)	-	-	Structure containing the DEM model. Different choices are posible depending on the DEM model.

Table 162: DEM Configuration File. ACE Model

Tag name	type	Attribute	C Format	Description
Directory	string	-	%s	Directory where the DEM files are placed. It can be an absolute or a relative path. All the files are assumed to be in the same directory. The filenames for DEM files should follow the following convention: xx{N/S}yyy{E/W}.GETASSE30 (where xx is the latitude and yyy the longitude of the southern western point in the file and N, S E and W stands for north, south, east and west respectively)
Interval_X	real	unit="*xxx*" Possible values: • deg • min • secs	%f	Angular interval between two consequtive points along the X-Axis in a DEM file.
Interval_Y	real	unit="*xxx*" Possible values: • deg • min • secs	%f	Angular interval between two consequtive points along the Y-Axis in a DEM file.
Num_Points_X	integer	-	%d	Number of points along the X-Axis in a DEM file
Num_Points_Y	integer	-	%d	Number of points along the Y-Axis in a DEM file

Table 162: DEM Configuration File. ACE Model

Tag name	type	Attribute	C Format	Description
Data_Type	string	-	%s	Data type for the altitude values in the dem. Posible values are: • int • long • float • double

8.11.2 File Example

```
<?xml version="1.0"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
         <File_Name>DEM_CONFIG_TEST_FILE</File_Name>
         <File_Description>DEM Configuration File</File_Description>
         <Notes/>
         <Mission></Mission>
         <File_Class>TEST</File_Class>
         <File_Type></File_Type>
         <Validity_Period>
            <Validity_Start>UTC=0000-00-00T00:00:00.000000</Validity_Start>
            <Validity_Stop>UTC=9999-99-99T99:99:99.999999</Validity_Stop>
         </Validity_Period>
         <File_Version>1</File_Version>
         <Source>
            <System>CFI Acceptance</System>
            <Creator></Creator>
            <Creator_Version></Creator_Version>
            <Creation_Date>UTC=2003-11-28T17:25:44</Creation_Date>
         </Source>
      </Fixed_Header>
      <Variable_Header/>
   </Earth_Explorer_Header>
   <Data_Block type="xml">
      <ACE_Model>
         <Directory>../../data/DEM</Directory>
         <Interval_X unit="secs">30</Interval_X>
         <Interval_Y unit="secs">30</Interval_Y>
         <Num_Points_X>1800</Num_Points_X>
         <Num_Points_Y>1800</Num_Points_Y>
         <Data_Type>float</Data_Type>
      </ACE_Model>
   </Data_Block>
</Earth_Explorer_File>
```

8.12 Swath Definition File

8.12.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty.
3. Data Block: It consists in a set of structures described in the tables below.

Table 163: Swath Definition File. Data Block

Tag name	type	Attribute	C Format	Description
Swath	Structure (see table 164)	-	-	Swath structure

Table 164: Swath Definition File. Swath

Tag name		type	Attribute	C Format	Description
Output_File_Description		string	-	%s	File Despcrition for the output swath template file
Output_File_Type		string	-	%s	File type for the output swath template file. It should have the fixed value “MPL_SWTREF”
Swath_Type		string	-	%s	Swath type. It can have one of the following values: • point • line • inertial
Num_Swath_Records		integer	-	%d	Number of points in the swath template file (>0)
Refraction		Structure (See table 165)	-	-	Refraction model structure
Choice between the following structures:	Point_Geometry	Structure (see table 166)	-	-	Point geometry
	Line_Geometry	Structure (see table 167)	-	-	Line geometry
	Limb_Geometry	Structure (see table 168)	-	-	Limb geometry
	Distance_Geometry	Structure (see table 170)	-	-	Distance geometry
Sat_Nominal_Att		Structure (see table 171)	-	-	Satellite Nominal Attitude initialization data
Sat_Att		Structure (see table 172)	-	-	Satellite Attitude initialization data
Instr_Att		Structure (see table 172)	-	-	Instrument Attitude initialization data

Table 165: Swath Definition File. Refraction

Tag name	type	Attribute	C Format	Description
Model	string	-	%s	Atmospheric refraction model. It can be one of: • NO_REF • STD_REF • USER_REF • PRED_REF
Freq	real	unit="MHz"	%f	Signal Frequency (≥ 0)

Table 166: Swath Definition File. Point Geometry

Tag name	type	Attribute	C Format	Description
Azimuth	real	unit="deg"	%f	Swath azimuth (-360, 360)
Elevation	real	unit="deg"	%f	Swath elevation (-90, 90)
Altitude	real	unit="m"	%f	Swath altitude

Table 167: Swath Definition File. Line Geometry

Tag name	type	Attribute	C Format	Description
Left_Pt	Point Geometry structure (see table 166)	-	-	Left point for line swath
Mid_Pt	Point Geometry structure (see table 166)	-	-	Middle point for line swath
Right_Pt	Point Geometry structure (see table 166)	-	-	Right point for line swath

Table 168: Swath Definition File. Limb Geometry

Tag name	type	Attribute	C Format	Description
Left_Pt	Inertial Point Geometry structure (table 169)	-	-	Left point for inertial swath
Mid_Pt	Inertial Point Geometry structure (table 169)	-	-	Middle point for inertial swath
Right_Pt	Inertial Point Geometry structure (table 169)	-	-	Right point for inertial swath

Table 169: Swath Definition File. Inertial Point Geometry

Tag name	type	Attribute	C Format	Description
Azimuth	real	unit="deg"	%f	Swath azimuth (-360, 360)

Table 169: Swath Definition File. Inertial Point Geometry

Tag name	type	Attribute	C Format	Description
Altitude	real	unit="m"	%f	Swath altitude

Table 170: Swath Definition File. Distance Geometry

Tag name	type	Attribute	C Format	Description
Azimuth	real	unit="deg"	%f	Swath azimuth (-360, 360)
Elevation	real	unit="deg"	%f	Swath elevation (-90, 90)
Altitude	real	unit="m"	%f	Swath altitude
Distance	real	unit="m"	%f	Distance

Table 171: Swath Definition File. Satellite Nominal Attitude

Tag name		type	Attribute	C Format	Description
Choice between one of the following tags:	None	Null (no value needed for this tag)	-	-	The satellite nominal attitude frame is not defined.
	AOCS_Model	integer	-	%d	AOCS model
	Parameter_Model	Structure (see table 173)	-	-	Attitude initialization with parameter model
	Harmonic_Model	Structure (see table 174)	-	-	Attitude initialization with harmonic model
	File_Model	Structure (see table 175)	-	-	Attitude initialization with a data file

Table 172: Swath Definition File. Satellite and Instrument Attitude

Tag name		type	Attribute	C Format	Description
Choice between one of the following tags:	None	Null (no value needed for this tag)	-	-	The attitude frame is not defined.
	Harmonic_Model	Structure (see table 174)	-	-	Attitude initialization with harmonic model
	File_Model	Structure (see table 175)	-	-	Attitude initialization with a data file
	Angle_Model	Structure (see table 176)	-	-	Attitude initialization with angles
	Matrix_Model	Structure (see table 177)	-	-	Attitude initialization with a Matrix

Table 173: Swath Definition File. Parameter Model

Tag name	type	Attribute	C Format	Description
Model	integer	-	%d	Parameter model

Table 173: Swath Definition File. Parameter Model

Tag name	type	Attribute	C Format	Description
List_of_Parameters	List of <*Parameter*> structures (see table 178)	count="*n*" where *n* is the number of elements in the list	-	List of parameters as used for the CFI function *xp_sat_nominal_att_init_model* (See [POINT_SUM])

Table 174: Swath Definition File. Harmonic Model

Tag name	type	Attribute	C Format	Description
Angle_Type	integer	-	%d	Angle type
List_of_Harmonics_Pitch	List of <*Harmonic_Pitch*> *structures* (see table 179)	count="*n*" where *n* is the number of elements in the list	-	List of harmonic pitch coeficients
List_of_Harmonic_Roll	List of <*Harmonic_Roll*> *structures* (see table 179)	count="*n*" where *n* is the number of elements in the list	-	List of harmonic roll coeficients
List_of_Harmonic_Yaw	List of <*Harmonic_Yaw*> *structures* (see table 179)	count="*n*" where *n* is the number of elements in the list	-	List of harmonic yaw coeficients
Offsets (only for Instr_Att element)	Structure (see table 181)	-	-	Offsets

Table 175: Swath Definition File. File Model

Tag name	type	Attribute	C Format	Description
List_of_Files	List of <*File*> (see *table 182*)	count="*n*" where *n* is the number of elements in the list	-	Attitude file list
Auxiliary_File (only for Sat_Att)	string	-	%s	Attitude auxiliary filename (complete path)
Time_Selection	Structure (see *table 183*)	-	-	It indicates the time window to be read from the attitude files.

Table 176: Swath Definition File. Angle Model

Tag name	type	Attribute	C Format	Description
Angle_1	real	unit="deg"	%f	Pitch Mispointing angle
Angle_2	real	unit="deg"	%f	Roll Mispointing angle
Angle_3	real	unit="deg"	%f	Yaw Mispointing angle
Offsets (only for Instr_Att element)	Structure (see table 181)	-	-	Offsets

Table 177: Swath Definition File. Matrix Model

Tag name	type	Attribute	C Format	Description
Row_1	Structure (see table 185)	-	-	Mispointing matrix first row
Row_2	Structure (see table 185)	-	-	Mispointing matrix second row
Row_3	Structure (see table 185)	-	-	Mispointing matrix third row
Offsets (only for Instr_Att element)	Structure (see table 181)	-	-	Offsets

Table 178: Swath Definition File. List_of_Parameters

Tag name	type	Attribute	C Format	Description
Parameter	string	-	%s	Parameter

Table 179: Swath Definition File. List_of_Harmonics_Pitch/Roll/Yaw

Tag name	type	Attribute	C Format	Description
Harmonic	Structure (see table 180)	-	-	Harmonic structure

Table 180: Swath Definition File. Harmonic

Tag name	type	Attribute	C Format	Description
Harmonic_Type	integer	-	%d	Harmonic type
Harmonic_Coef	real	-	%f	Harmonic coefficient

Table 181: Swath Definition File. Offsets

Tag name	type	Attribute	C Format	Description
Offset_X	real	-	%f	X Offset
Offset_Y	real	-	%f	Y Offset
Offset_Z	real	-	%f	Z Offset

Table 182: Swath Definition File. File

Tag name	type	Attribute	C Format	Description
File	string	-	-	Attitude filename (complete path)

Table 183: Swath Definition File. Time Selection

Tag name	type	Attribute	C Format	Description
Select_File	Null (no value needed for this tag)	-	-	The whole files will be read from the files
Time_Window	Structure (see table 184)	-	-	A time window will be read from the files

Table 184: Swath Definition File. Time_Window

Tag name	type	Attribute	C Format	Description
Time_0	real	-	%f	Start time
Time_1	real	-	%f	Stop time

Table 185: Swath Definition File. Row

Tag name	type	Attribute	C Format	Description
Column_1	real	-	%f	Matrix element in the first column
Column_2	real	-	%f	Matrix element in the second column
Column_3	real	-	%f	Matrix element in the third column

8.12.2 File Example

```
<?xml version="1.0"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
         <File_Name>SWATH_DEF_FILE.XML</File_Name>
         <File_Description>Swath Definition File</File_Description>
         <Notes/>
         <Mission>XXXXX</Mission>
         <File_Class>TEST</File_Class>
         <File_Type>MPL_SW_DEF</File_Type>
         <Validity_Period>
            <Validity_Start>UTC=0000-00-00T00:00:00.000000</Validity_Start>
            <Validity_Stop>UTC=9999-99-99T99:99:99.999999</Validity_Stop>
         </Validity_Period>
         <File_Version>1</File_Version>
         <Source>
            <System>CFI Acceptance</System>
            <Creator></Creator>
            <Creator_Version></Creator_Version>
            <Creation_Date>UTC=2003-11-28T17:25:44</Creation_Date>
         </Source>
      </Fixed_Header>
      <Variable_Header/>
   </Earth_Explorer_Header>
   <Data_Block type="xml">
      <Swath>
```

```
        <Output_File_Description>MERIS</Output_File_Description>
        <Output_File_Type>MERIS__501</Output_File_Type>
        <Swath_Type>line</Swath_Type>
        <Num_Swath_Records>1200</Num_Swath_Records>
        <Refraction>
          <Model>NO_REF</Model>
          <Freq unit="MHz">000440000000</Freq>
        </Refraction>
        <Line_Geometry>
          <Left_Pt>
            <Azimuth unit="deg">+270.000000</Azimuth>
            <Elevation unit="deg">+055.750000</Elevation>
            <Altitude unit="m">+000000.000</Altitude>
          </Left_Pt>
          <Mid_Pt>
            <Azimuth unit="deg">+090.000000</Azimuth>
            <Elevation unit="deg">+090.000000</Elevation>
            <Altitude unit="m">+000000.000</Altitude>
          </Mid_Pt>
          <Right_Pt>
            <Azimuth unit="deg">+090.000000</Azimuth>
            <Elevation unit="deg">+055.750000</Elevation>
            <Altitude unit="m">+000000.000</Altitude>
          </Right_Pt>
        </Line_Geometry>

        <Sat_Nominal_Att>
          <Parameter_Model>
            <Model>1</Model>
            <List_of_Parameters count="3">
              <Parameter>-000.167200</Parameter>
              <Parameter>+000.050100</Parameter>
              <Parameter>+003.928400</Parameter>
            </List_of_Parameters>
          </Parameter_Model>
        </Sat_Nominal_Att>
        <Sat_Att>
          <Angle_Model>
            <Angle_1>0</Angle_1>
            <Angle_2>0</Angle_2>
            <Angle_3>0</Angle_3>
          </Angle_Model>
        </Sat_Att>
        <Instr_Att>
          <None></None>
        </Instr_Att>
      </Swath>
   </Data_Block>
</Earth_Explorer_File>
```

8.13 Swath Template File

8.13.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: It consists in a set of structures described in the tables below.

Table 186: Swath Template File. Variable_Header

Tag name		type	Attribute	C Format	Description
Swath_Def_File		string	-	%s	Swath definition file used for generating the file
Swath_Type		string. It can have one of the following values: • point • line • inertial	-	%s	Swath type
One of the following options:	Orbit_Geometry	Structure (see table 186)	-	-	Set of orbital parameters
	Orbit_State_Vector	Structure (see table 187)	-	-	Orbit state vector
Time_Step		real	unit="s"	%f	Seconds between two swath points
One of the following options:	Point_Altitude	real	unit="m"	%f	Altitude for the point swath
	Line_Altitude	Structure (see table 189)	-	-	Altitudes for the line swath
Refraction		Structure (see table 190)	-	-	Refraction data

Table 187: Swath Template File. Orbit_Geometry

Tag name	type	Attribute	C Format	Description
Repeat_Cycle	real	unit="day"	%f	Repeat cycle in days
Cycle_Length	real	unit="orbit"	%f	Cycle length in orbits
MLST_Drift	real	unit="s/day"	%f	Mean local solar time drift

Table 188: Swath Template File. Orbit_State_Vector

Tag name	type	Attribute	C Format	Description
Absolute_Orbit	integer	-	%d	Orbit number for the swath template file
Pos_X	real	unit="m"	%f	Position in X coordinate (meters)
Pos_Y	real	unit="m"	%f	Position in Ycoordinate (meters)
Pos_Z	real	unit="m"	%f	Position in Z coordinate (meters)
Vel_X	real	unit="m/s"	%f	Velocity in X coordinate

Table 188: Swath Template File. Orbit_State_Vector

Tag name	type	Attribute	C Format	Description
Vel_Y	real	unit="m/s"	%f	Velocity in Y coordinate
Vel_Z	real	unit="m/s"	%f	Velocity in Z coordinate

Table 189: Swath Template File. Line_Altitude

Tag name	type	Attribute	C Format	Description
Left_Altitude	real	unit="m"	%f	Swath altitude for the left point
Mid_Altitude	real	unit="m"	%f	Swath altitude for the middle point
Right_Altitude	real	unit="m"	%f	Swath altitude for the right point

Table 190: Swath Template File. Refraction

Tag name	type	Attribute	C Format	Description
Model	string	-	%s	Atmospheric refraction model. It can be one of: • NO_REF • STD_REF • USER_REF • PRED_REF
Freq	real	unit="MHz"	%f	Signal Frequency (≥ 0)

3. Data Block: It consists in a set of structures described in the tables below.

Table 191: Swath Template File. Data_Block

Tag name		type	Attribute	C Format	Description
One of the following options:	List_of_Point_Swaths	List of <*Point_Swath*> (See table 192)	count="*n*" where *n* is the number of elements in the list	-	List of points in the swath
	List_of_Line_Swaths	List of <*Line_Swath*> (See *table 193*)	count="*n*" where *n* is the number of elements in the list	-	List of points in the swath

Table 192: Swath Template File. Point_Swath

Tag name	type	Attribute	C Format	Description
Long	real	unit="deg"	%f	Longitude of the point
Lat	real	unit="deg"	%f	Longitude of the point

Table 193: Swath Template File. Line_Swath

Tag name	type	Attribute	C Format	Description
Left_Pt	Point Swath Structure (See table 192)	-	-	Left point in a line swath
Mid_Pt	Point Swath Structure (See table 192)	-	-	Middle point in a line swath
Right_Pt	Point Swath Structure (See table 192)	-	-	Right point in a line swath

8.13.2 File Example

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Earth_Explorer_File>
  <Earth_Explorer_Header>
    <Fixed_Header>
         <File_Name>LINE_SWATH_FILE.XML</File_Name>
         <File_Description>Swath Template File</File_Description>
         <Notes/>
         <Mission>XXXXX</Mission>
         <File_Class>TEST</File_Class>
         <File_Type>MPL_SWTDEF</File_Type>
         <Validity_Period>
            <Validity_Start>UTC=0000-00-00T00:00:00.000000</Validity_Start>
            <Validity_Stop>UTC=9999-99-99T99:99:99.999999</Validity_Stop>
         </Validity_Period>
         <File_Version>1</File_Version>
         <Source>
            <System>CFI Acceptance</System>
            <Creator></Creator>
            <Creator_Version></Creator_Version>
            <Creation_Date>UTC=2005-07-09T09:25:44</Creation_Date>
         </Source>
    </Fixed_Header>
    <Variable_Header>
      <Swath_Def_File></Swath_Def_File>
      <Swath_Type>line</Swath_Type>
      <Orbit_Geometry>
        <Repeat_Cycle unit="day">35</Repeat_Cycle>
        <Cycle_Length unit="orbit">501</Cycle_Length>
        <MLST_Drift unit="s/day">+000.000000</MLST_Drift>
      </Orbit_Geometry>
      <Time_Step unit="s">5.029940120</Time_Step>
      <Line_Altitude>
        <Left_Altitude unit="m">+000000.000</Left_Altitude>
        <Mid_Altitude unit="m">+000000.000</Mid_Altitude>
        <Right_Altitude unit="m">+000000.000</Right_Altitude>
      </Line_Altitude>
      <Refraction>
        <Model>NO_REF</Model>
        <Freq unit="MHz">0440000000</Freq>
      </Refraction>
    </Variable_Header>
  </Earth_Explorer_Header>
  <Data_Block type="xml">
    <List_of_Line_Swaths count="1201">
```

```
      <Line_Swath>
        <Left_Pt>
          <Long unit="deg">-004.850820</Long>
          <Lat unit="deg">-001.078218</Lat>
        </Left_Pt>
        <Mid_Pt>
          <Long unit="deg">-000.000000</Long>
          <Lat unit="deg">-000.000000</Lat>
        </Mid_Pt>
        <Right_Pt>
          <Long unit="deg">+004.850820</Long>
          <Lat unit="deg">+001.078218</Lat>
        </Right_Pt>
      </Line_Swath>
      <Line_Swath>
        <Left_Pt>
          <Long unit="deg">-004.915732</Long>
          <Lat unit="deg">-000.780732</Lat>
        </Left_Pt>
        <Mid_Pt>
          <Long unit="deg">-000.065595</Long>
          <Lat unit="deg">+000.298507</Lat>
        </Mid_Pt>
        <Right_Pt>
          <Long unit="deg">+004.785485</Long>
          <Lat unit="deg">+001.375591</Lat>
        </Right_Pt>
      </Line_Swath>
      [...]
      <Line_Swath>
        <Left_Pt>
          <Long unit="deg">329.999479</Long>
          <Lat unit="deg">-001.078218</Lat>
        </Left_Pt>
        <Mid_Pt>
          <Long unit="deg">-025.149701</Long>
          <Lat unit="deg">+000.0000000</Lat>
        </Mid_Pt>
        <Right_Pt>
          <Long unit="deg">-20.2988805</Long>
          <Lat unit="deg">+001.078218</Lat>
        </Right_Pt>
      </Line_Swath>
    </List_of_Line_Swaths>
  </Data_Block>
</Earth_Explorer_File>
```

8.14 Zone Database File

8.14.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 194: Zone Database File. Data_Block

Tag name	type	Attribute	C Format	Description
List_of_Zones	List of <Zone> Structures (See table 195)	count="*n*" where *n* is the number of elements in the list	-	List of zones

Table 195: Zone Database File. Zone

Tag name	type	Attribute	C Format	Description
Zone_Id	string	-	%8s	Zone name
Zone_Description	string	-	%s	Zone description
Surface	string	-	%s	Type of surface
Projection	string	-	%s	Projection
Creator	string	-	%s	Cretator name
List_of_Polygon_Pts	list of structures <Polygon_Pt> (See table 196)	count="*n*" where *n* is the number of elements in the list	-	List of points defining the zone.
Diameter	real	unit="m"	%f	Diameter of the zone if the list of polygon points is emtpy.

Table 196: Zone Database File. Polygon_Pt

Tag name	type	Attribute	C Format	Description
Long	real	unit="deg"	%f	longitude of the point (-360, 360)
Lat	real	unit="deg"	%f	latitude of the point (-90, 90)

8.14.2 File Example

```
<?xml version="1.0"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
         <File_Name>ZONE_FILE.XML</File_Name>
         <File_Description>Zone Database File</File_Description>
         <Notes/>
```

```
        <Mission>XXXXX</Mission>
        <File_Class>TEST</File_Class>
        <File_Type></File_Type>
        <Validity_Period>
           <Validity_Start>UTC=0000-00-00T00:00:00.000000</Validity_Start>
           <Validity_Stop>UTC=9999-99-99T99:99:99.999999</Validity_Stop>
        </Validity_Period>
        <File_Version>1</File_Version>
        <Source>
           <System>CFI Acceptance</System>
           <Creator></Creator>
           <Creator_Version></Creator_Version>
           <Creation_Date>UTC=2003-11-28T17:25:44</Creation_Date>
        </Source>
     </Fixed_Header>
     <Variable_Header/>
  </Earth_Explorer_Header>
  <Data_Block type="xml">
    <List_of_Zones count="5">
     <Zone>
       <Zone_Id>ZMIK____</Zone_Id>
       <Zone_Description></Zone_Description>
       <Surface></Surface>
       <Projection>ANY</Projection>
       <Creator>TEST DATA</Creator>
       <List_of_Polygon_Pts count="003">
         <Polygon_Pt>
           <Long unit="deg">+000.000000</Long>
           <Lat unit="deg">+000.000000</Lat>
         </Polygon_Pt>
         <Polygon_Pt>
           <Long unit="deg">+000.000000</Long>
           <Lat unit="deg">+000.000000</Lat>
         </Polygon_Pt>
         <Polygon_Pt>
           <Long unit="deg">+000.000000</Long>
           <Lat unit="deg">+000.000000</Lat>
         </Polygon_Pt>
       </List_of_Polygon_Pts>
       <Diameter unit="m">+0000000.000</Diameter>
     </Zone>

     <Zone>
       <Zone_Id>SEGMENT_</Zone_Id>
       <Zone_Description></Zone_Description>
       <Surface></Surface>
       <Projection>ANY</Projection>
       <Creator>TEST DATA</Creator>
       <List_of_Polygon_Pts count="002">
         <Polygon_Pt>
           <Long unit="deg">+000.000000</Long>
           <Lat unit="deg">+030.000000</Lat>
         </Polygon_Pt>
         <Polygon_Pt>
           <Long unit="deg">+150.000000</Long>
           <Lat unit="deg">+020.000000</Lat>
         </Polygon_Pt>
       </List_of_Polygon_Pts>
       <Diameter unit="m">+0000000.000</Diameter>
     </Zone>
```

```
      <Zone>
        <Zone_Id>POINT_DI</Zone_Id>
        <Zone_Description></Zone_Description>
        <Surface></Surface>
        <Projection>ANY</Projection>
        <Creator>TEST DATA</Creator>
        <List_of_Polygon_Pts count="001">
          <Polygon_Pt>
            <Long unit="deg">+000.000000</Long>
            <Lat unit="deg">+030.000000</Lat>
          </Polygon_Pt>
        </List_of_Polygon_Pts>
        <Diameter unit="m">+0100000.000</Diameter>
      </Zone>

      <Zone>
        <Zone_Id>POINT___</Zone_Id>
        <Zone_Description></Zone_Description>
        <Surface></Surface>
        <Projection>ANY</Projection>
        <Creator>TEST DATA</Creator>
        <List_of_Polygon_Pts count="001">
          <Polygon_Pt>
            <Long unit="deg">+002.278785</Long>
            <Lat unit="deg">-067.992416</Lat>
          </Polygon_Pt>
        </List_of_Polygon_Pts>
        <Diameter unit="m">+0000000.000</Diameter>
      </Zone>

      <Zone>
        <Zone_Id>Z_WORLD_</Zone_Id>
        <Zone_Description></Zone_Description>
        <Surface></Surface>
        <Projection>ANY</Projection>
        <Creator>TEST DATA</Creator>
        <List_of_Polygon_Pts count="000">
        </List_of_Polygon_Pts>
        <Diameter unit="m">+0000000.000</Diameter>
      </Zone>
    </List_of_Zones>
  </Data_Block>
</Earth_Explorer_File>
```

8.15 Station Database File

8.15.1 Format

1. Fixed Header: For the fixed header format, refer to [EE_FMT] section 7.1
2. Variable Header: Empty
3. Data Block: It consists in a set of structures described in the tables below:

Table 197: Station Database File. Data Block

Tag name	type	Attribute	C Format	Description
Station_Id	string	-	%8s	Station name
Descriptor	string	-	%s	Station description
Antenna	string	-	%s	Antenna band
Frequency (optional)	real	unit="Hz"	%f	Frequency
Purpose	string	-	%s	Purpose
Type	string	-	%s	
Location	Structure (see table 198)	-	-	Station location
Default_El	real	unit="deg"	%f	Default elevation
List_of_Mask_Points	list of <Mask_Point> structures (see table 199)	count="*n*" where *n* is the number of elements in the list	-	Mask points

Table 198: Station Database File. Location

Tag name	type	Attribute	C Format	Description
Long	real	unit="deg"	%f	longitude
Lat	real	unit="deg"	%f	Latitude
Alt	real	unit="deg"	%f	Altitude

Table 199: Station Database File. Mask_Point

Tag name	type	Attribute	C Format	Description
Az	real	unit="deg"	%f	Azimuth
El	real	unit="deg"	%f	Elevation

8.15.2 File Example

```
<?xml version="1.0"?>
<Earth_Explorer_File>
   <Earth_Explorer_Header>
      <Fixed_Header>
```

```
        <File_Name>STATION_FILE.XML</File_Name>
        <File_Description>Station Database File</File_Description>
        <Notes/>
        <Mission>XXXXX</Mission>
        <File_Class>TEST</File_Class>
        <File_Type></File_Type>
        <Validity_Period>
           <Validity_Start>UTC=0000-00-00T00:00:00.000000</Validity_Start>
           <Validity_Stop>UTC=9999-99-99T99:99:99.999999</Validity_Stop>
        </Validity_Period>
        <File_Version>1</File_Version>
        <Source>
           <System>CFI Acceptance</System>
           <Creator></Creator>
           <Creator_Version></Creator_Version>
           <Creation_Date>UTC=2003-11-28T17:25:44</Creation_Date>
        </Source>
     </Fixed_Header>
     <Variable_Header/>
  </Earth_Explorer_Header>
  <Data_Block type="xml">
    <List_of_Ground_Stations count="n">
     <Ground_Station>
       <Station_id>GKIRUNBX</Station_id>
       <Descriptor>Kiruna (SWEDEN)</Descriptor>
       <Antenna>X-BAND</Antenna>
       <Purpose>GLOBAL + REGIONAL</Purpose>
       <Type></Type>
       <Location>
         <Long unit="deg">+020.964100</Long>
         <Lat unit="deg">+067.857000</Lat>
         <Alt unit="m">+0362.000</Alt>
       </Location>
       <Default_El unit="deg">+000.000000</Default_El>
       <List_of_Mask_Points count="073">
         <Mask_Point>
           <Az unit="deg">+000.000000</Az>
           <El unit="deg">+001.250000</El>
         </Mask_Point>
         <Mask_Point>
           <Az unit="deg">+004.000000</Az>
           <El unit="deg">+001.150000</El>
         </Mask_Point>
         <Mask_Point>
           <Az unit="deg">+010.000000</Az>
           <El unit="deg">+001.270000</El>
         </Mask_Point>
         [...]
         <Mask_Point>
           <Az unit="deg">+360.000000</Az>
           <El unit="deg">+001.250000</El>
         </Mask_Point>
       </List_of_Mask_Points>
     </Ground_Station>
     <Ground_Station>
       <Station_id>GAREA__D</Station_id>
       <Descriptor>AREQUIPA (PEROU)</Descriptor>
       <Antenna>DORIS </Antenna>
       <Purpose></Purpose>
       <Type></Type>
       <Location>
```

```
          <Long unit="deg">-071.500000</Long>
          <Lat unit="deg">-016.470000</Lat>
          <Alt unit="m">+2494.000</Alt>
        </Location>
        <Default_El unit="deg">+012.000000</Default_El>
        <List_of_Mask_Points count="000">
        </List_of_Mask_Points>
      </Ground_Station>
      [...]
    </List_of_Ground_Stations>
   </Data_Block>
</Earth_Explorer_File>
```

9 LIBRARY PRECAUTIONS

The following precaution shall be taking into account when using EXPLORER_DATA_HANDLING library:

- None

10 KNOWN PROBLEMS

The following precautions shall be taken into account when using the CFI software libraries:

Table 200: Known problems

CFI library	Problem	Work around solution
xd_xml_validate	Funtion is not currently available	-