en

Earth Observation Mission CFI Software

EO_VISIBILITY SOFTWARE USER MANUAL

Code:	EO-MA-DMS-GS-0006
Issue:	4.8
Date:	29/10/2014

	Name	Function	Signature
Prepared by:	José Antonio González Abeytua	Project Manager	
	Juan José Borrego Bote	Project Engineer	
	Carlos Villanueva Muñoz	Project Engineer	
	Rubén Castro	Project Engineer	
Checked by:	José Antonio González Abeytua	Project Manager	
Approved by:	José Antonio González Abeytua	Project Manager	

DEIMOS Space S.L.U
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:	LibreOffice 3.6
Archive Code:	P/SUM/DMS/01/026-029
Electronic file name:	eo-ma-dms-gs-006-21

DOCUMENT STATUS LOG

Issue	Change Description	Date	Approval
1.0	Unreleased	19/06/02	
2.0	Complete document	29/11/02	
2.1	Maintenance release with the following main changes: • xv_multizones_vis_time added. • xv_multistation_vis_time added. • xv_time_segment_mapping added. • xv_orbit_extra added.	13/05/03	
2.2	Maintenance release	30/09/03	
2.2.2	Small interface chance in xv_time_segments_delta and xv_orbit_extra	26/04/04	
3.0	New initialisation strategy and interfaces.	21/07/04	
3.1	New features for xv_zone_vis_time function: • Use of Predicted Orbit/Orbit event files. • Use of Swath Definition files.	13/10/04	
3.2	Maintenance release	15/11/04	
3.3	New features: • Use of Predicted Orbit/Orbit event files for all visibility functions. • Use of Swath Definition files for all visibility functions. • ENVISAT ASCII files are no longer supported	11/07/05	
3.4	• Maintenance release. • gen_swath executable moved to this library. • Changes in the xv_swath_pos interface	18/11/05	
3.5	Maintenance release.	26/05/06	
3.6	Maintenance release.	24/11/06	

3.7	• Maintenance release. • New features: - xv_gen_scf - expcfi_check_libs - xv_zone_vis_time_no_file - xv_station_vis_time_no_file - xv_gen_swath_no_file - library version for Mac OS X on Intel (32 and 64 bits)	13/07/07	
3.7.2	• Maintenance release. • New features: - Curved and closed swaths for xv_zone_vis_time	31/07/08	
4.0	Maintenance release.	19/01/09	
4.1	• Maintenance release • New features: - AOS/LOS mask mode from Ground Station DB 1.4 file - Visibility across orbital changes	07/05/10	
4.2	• Maintenance release • New features: - Support of visibility functions with TLE and precise propagator		
4.3	• Maintenance release		
4.4	• Maintenance release • New features: - Function xv_sc_vis_time	05/07/12	
4.5	• Maintenance release	01/03/13	
4.6	• Maintenance release • New features: - Support for new swath ID. - New functions: xv_zonevistime_compute xv_stationvistime_compute	03/10/13	

	xv_swathpos_compute xv_tiesegments_xxx_compute		

4.7	• Maintenance release
	• New features:
	  • New function xv_zonevistime_coverage
	  • Support for satellites Jason-CS, MetOp-SG and Sentinel_5P
		28/03/2014	

4.8	• Maintenance release	29/10/2014	

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 SCOPE

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

2 ACRONYMS, NOMENCLATURE AND TERMINOLOGY

2.1 Acronyms

ANX	Ascending Node Crossing
AOCS	Attitude and Orbit Control Subsystem
CFI	Customer Furnished Item
EF	Earth Fixed reference frame
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
FOS	Flight Operations Segment
GS	Ground Station
OSF	Orbit Scenario File
SCF	Swath Control File
SDF	Swath Definition File
SRAR	Satellite Relative Actual Reference
SSP	Sub-Satellite Point
STF	Swath Template File
SUM	Software User Manual
TOD	True of Date reference frame
UTC	Universal Time Coordinated
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)

2.3 Note on Terminology

In order to keep compatibility with legacy CFI libraries, the Earth Observation Mission CFI Software makes use of terms that are linked with missions already or soon in the operational phase like the Earth Explorers.

This may be reflected in the rest of the document when examples of Mission CFI Software usage are proposed or description of Mission Files is given.

3 APPLICABLE AND REFERENCE DOCUMENTS

3.1 Applicable Documents

No applicable documents.

3.2 Reference Documents

[GEN_SUM] Earth Observation Mission CFI Software. General Software User Manual. EO-MA- DMS-GS-0002.

[F_H_SUM] Earth Observation Mission CFI Software. EO_FILE_HANDLING Software User Manual. EO-MA-DMS-GS-0008.

[D_H_SUM] Earth Observation Mission CFI Software. EO_DATA_HANDLING Software User Manual. EO-MA-DMS-GS-007.

[LIB_SUM] Earth Observation Mission CFI Software. EO_LIB Software User Manual. EO-MA-DMS-GS-003.

[ORBIT_SUM] Earth Observation Mission CFI Software. EO_ORBIT Software User Manual. EO-MA-DMS-GS-0004.

[POINT_SUM] Earth Observation Mission CFI Software. EO_POINTING Software User Manual. EO-MA-DMS-GS-0005.

[FORMATS] Earth Explorer File Format Guidelines. CS-TN-ESA-GS-0148.

4 INTRODUCTION

4.1 Functions Overview

This software library contains the CFI functions required to compute time segments at which an Earth Observation satellite, or one of its instruments is in view of various targets:

- zones (defined as polygons or circles, on the earth ellipsoid or at a given altitude)
- ground stations
- data relay satellites
- stars

This library is to be used for planning of Earth Observation operations. It includes, the following CFI functions:

- **xv_stationvistime_compute:** compute visibility time segments for one or several ground stations.
- **xv_sc_vis_time**: computes visibility time segments for a target satellite
- **xv_zonevistime_compute**: compute visibility time segments for an instrument swath in visibility of one or several zones.
- **xv_swathpos_compute**: computes location of a swath at a given time (additional routine to help refine the results of **xv_zonevistime_compute**)
- **xv_star_vis_time**: computes visibility time segments for a star.
- **xv_gps_vis_time:** computes visibility time segments for a gps constellation.
- **xv_gen_swath** and **xv_gen_swath_no_file** generate the instrument swath template file for a given satellite, instrument mode and orbit.
- **xv_gen_scf** generates a swath control file for the ESOV tool.
- Time Segments Manipulation Routines:
  - **xv_timesegments_compute_not:** returns the complement of 1 vector of time segments.
  - **xv_timesegments_compute_and:** returns the intersection segments from 2 vectors of time segments.
  - **xv_timesegments_compute_or:** returns the joined segments from 2 vectors of time segments
  - **xv_timesegments_compute_delta:** add or subtract time durations at the beginning and end of each time segment in a vector.
  - **xv_timesegments_compute_sort:** returns the vector of time segments sorted according to absolute or relative orbits.
  - **xv_timesegments_compute_merge:** merges all the overlapped segments in a list.
  - **xv_timesegments_compute_mapping:** returns a subset of the time segments vector, such that this subset covers entirely a zone or line swath.

Several files are required to operate properly the above functions:

- Orbit Scenario File (all functions)
- Swath Template Files (**xv_stationvistime_compute, xv_zonevistime_compute, xv_swathpos_compute**)
- Ground Stations Database File (**xv_stationvistime_compute**)
- (optionally) Zones Database File (**xv_zonevistime_compute**)
- (optionally) Star Database File (**xv_star_vis_time**)

Note that all the above routines use orbit-relative time parameters (i.e. the time parameters are represented as orbit number + time since ascending node). Two functions from EO_ORBIT will be very useful to process the input/outputs:

- **xo_time_to_orbit**: converts from TAI/UTC/UT1 time to orbit-relative time
- **xo_orbit_to_time**: converts from orbit-relative time to TAI/UTC/UT1 time

Since version of EOCFI 4.6, some of the visibility functions have been substituted by newer functions, which have an equivalent functionality. The older versions have been declared deprecated, although they can still be used; anyway, it is recommended to use the new functions. In the table 1 you can find the correspondence between the old (deprecated) functions and the new ones which have an equivalent functionality.

Table 1: Correspondence of current functions and deprecated functions

Current function	Corresponding deprecated function(s)
xv_stationvistime_compute	xv_station_vis_time xv_station_vis_time_no_file xv_multistations_vis_time
xv_swathpos_compute	xv_swath_pos
xv_timesegments_compute_and	xv_time_segments_and
xv_timesegments_compute_or	xv_time_segments_delta
xv_timesegments_compute_not	xv_time_segments_mapping
xv_timesegments_compute_sort	xv_time_segments_merge
xv_timesegments_compute_merge	xv_time_segments_not
xv_timesegments_compute_delta	xv_time_segments_or
xv_timesegments_compute_mapping	xv_time_segments_sort
xv_zonevistime_compute	xv_zone_vis_time xv_zone_vis_time_no_file xv_multizones_vis_time

4.2 Calling Sequence

An overview of the data flow is presented in Figure 1,

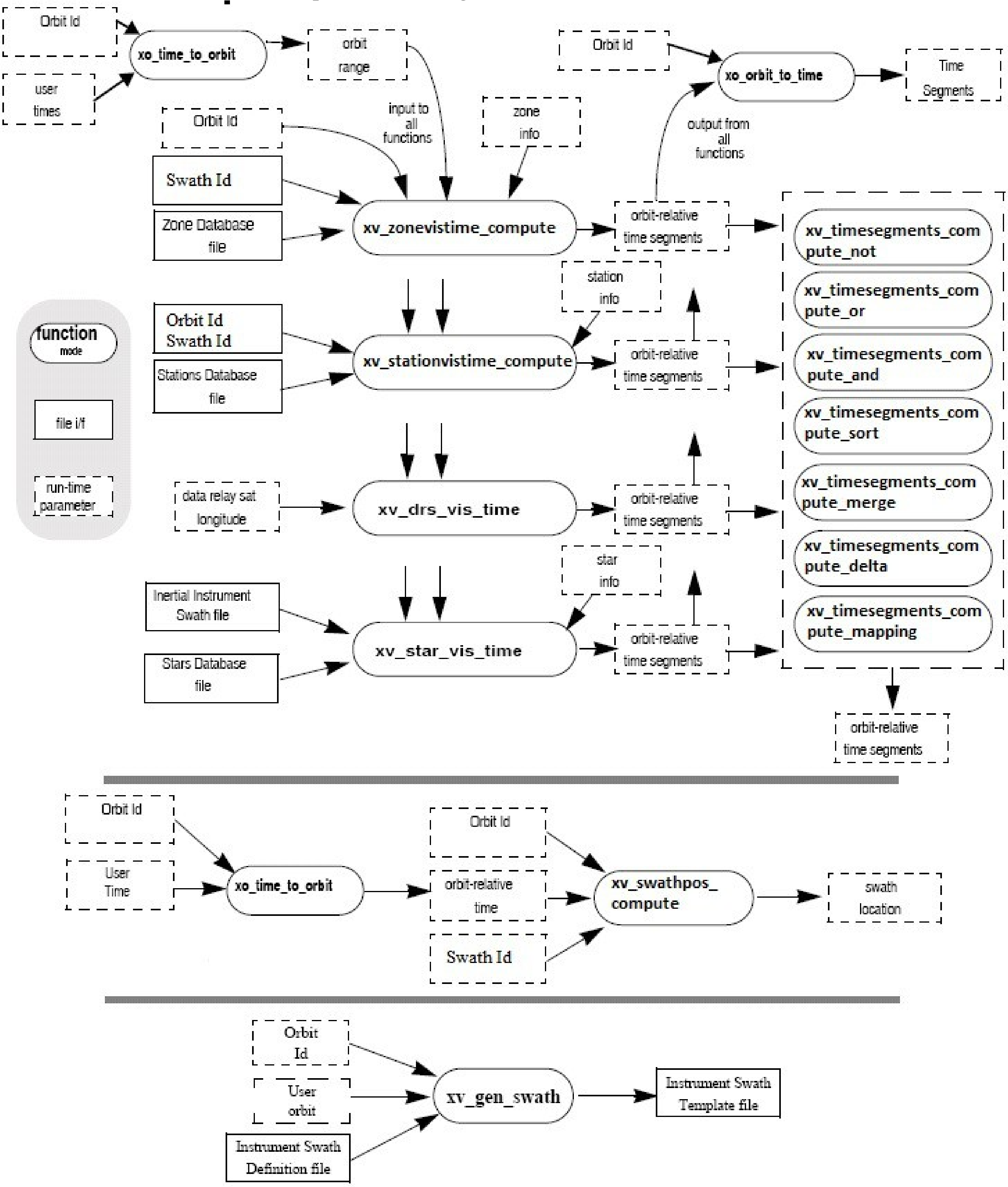


Figure 1: EO_VISIBILITY Data Flow

5 LIBRARY INSTALLATION

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

Note that example data files are provided with this CFI.

6 LIBRARY USAGE

The EO_VISIBILITY software library has the following dependencies:

- Other EOCFI libraries:
  - EO_FILE_HANDLING (See [F_H_SUM]).
  - EO_DATA_HANDLING (See [D_H_SUM]).
  - EO_LIB (See [LIB_SUM]).
  - EO_ORBIT (See [ORBIT_SUM]).
  - EO_POINTING (See [POINT_SUM]).
- Third party libraries:
  - POSIX thread library: libpthread.so (Note: this library is normally pre-installed in Linux and MacOS platforms. For Windows platforms, pthread.lib is included in the distribution package, with license LGPL);
  - GEOTIFF, TIFF, PROJ, LIBXML2 libraries (these libraries are included in the distribution package. Their usage terms and conditions are available in the file "TERMS_AND_CONDITIONS.TXT" which is part of the distribution package).

In order to improve run-time performance, Some pointing functions (see section 7.88, 7.90, 7.92 of [POINT_SUM]) perform their computations in multi-threading mode.
The multi-threading code of the Pointing functions uses the OpenMP API (see http://en.wikipedia.org/wiki/OpenMP).
OpenMP is not supported in the clang compiler, therefore such functions work in single-thread mode in MacOS.

The following is required to compile and link a Software application that uses the EO_VISIBILITY software library functions (it is assumed that the required EOCFI and third-part libraries are located in directory *cfi_lib_dir* and the required header files are located in *cfi_include*, see [GEN_SUM] for installation procedures):

1) include the following header files in the source code:

   - explorer_visibility.h (for a C application)

2) use the following compile and link options:

   Linux platforms:

   -I*cfi_include_dir* -L*cfi_lib_dir* -lexplorer_visibility

   -lexplorer_pointing -lexplorer_orbit -lexplorer_lib - lexplorer_data_dandling -lexplorer_file_handling -lgeotiff -ltiff -lproj -lxml2 -lm -lc -lpthread -fopenmp

   MacOS platforms (openmp is not supported)

   -I*cfi_include_dir* -L*cfi_lib_dir* -lexplorer_visibility

   -lexplorer_pointing -lexplorer_orbit -lexplorer_lib - lexplorer_data_dandling -lexplorer_file_handling -lgeotiff -ltiff -lproj -lxml2 -lm -lc -lpthread

Windows platforms:

/I *"cfi_include_dir"* /libpath:*"cfi_lib_dir"* libexplorer_visibility.lib

libexplorer_pointing.lib libexplorer_orbit.lib libexplorer_lib.lib libexplorer_data_handling.lib libexplorer_file_handling.lib libgeotiff.lib libtiff.lib libproj.lib libxml2.lib pthread.lib /openmp

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

To avoid problems in linking a user application with the `EO_VISIBILITY` 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 `XV_` or `xv_`.

This is summarized in Table 2.

Table 2: CFI functions included within EO_VISIBILITY library

Function Name	Enumeration value	long
Main CFI Functions		
xv_zone_vis_time xv_zone_vis_time_no_file	XV_ZONE_VIS_TIME_ID	0
xv_station_vis_time xv_station_vis_time_no_file	XV_STATION_VIS_TIME_ID	1
xv_sc_vis_time	XV_SC_VIS_TIME_ID	2
xv_swath_pos_id	XV_SWATH_POS_ID	3
xv_star_vis_time	XV_STAR_VIS_TIME_ID	4
xv_multizones_vis_time	XV_MULTIZONES_VIS_TIME_ID	5
xv_multistations_vis_time	XV_MULTISTATIONS_VIS_TIME_ID	6
xv_time_segments_not	XV_TIME_SEGMENTS_NOT_ID	7
xv_time_segments_or	XV_TIME_SEGMENTS_OR_ID	8
xv_time_segments_and	XV_TIME_SEGMENTS_AND_ID	9
xv_time_segments_sort	XV_TIME_SEGMENTS_SORT_ID	10
xv_time_segments_merge	XV_TIME_SEGMENTS_MERGE_ID	11
xv_time_segments_delta	XV_TIME_SEGMENTS_DELTA_ID	12
xv_time_segments_mapping	XV_TIME_SEGMENTS_MAPPING_ID	13
xv_orbit_extra	XV_ORBIT_EXTRA_ID	14
xv_gen_scf	XV_GEN_SCF_ID	15
xv_gen_swath xv_gen_swath_no_file	XV_GEN_SWATH_ID	16
xv_swath_id_init	XV_SWATH_ID_INIT_ID	17

xv_swath_id_close	XV_SWATH_ID_CLOSE_ID	18
xv_zonevistime_compute	XV_ZONEVISTIME_COMPUTE_ID	19
xv_stationvistime_compute	XV_STATIONVISTIME_COMPUTE_ID	20
xv_swathpos_compute	XV_SWATHPOS_COMPUTE_ID	21
xv_timesegments_compute_not	XV_TIMESEGMENTS_COMPUTE_NOT_ID	22
xv_timesegments_compute_or	XV_TIMESEGMENTS_COMPUTE_OR_ID	23
xv_timesegments_compute_and	XV_TIMESEGMENTS_COMPUTE_AND_ID	24
xv_timesegments_compute_sort	XV_TIMESEGMENTS_COMPUTE_SORT_ID	25
xv_timesegments_compute_merge	XV_TIMESEGMENTS_COMPUTE_MERGE_ID	26
xv_timesegments_compute_delta	XV_TIMESEGMENTS_COMPUTE_DELTA_ID	27
xv_timesegments_compute_mapping	XV_TIMESEGMENTS_COMPUTE_MAPPING_ID	28
Error Handling Functions		
xv_verbose	not applicable	
xv_silent		
xv_get_code		
xv_get_msg		
xv_print_msg		

Notes about the table:

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

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 (XV_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 EO_VISIBILITY routines, as shown in the table below. The enumerations presented in [GEN_SUM] are also applicable.

Table 3: Some enumerations within EO_VISIBILITY library

Input	Description	Enumeration value	Long
Orbit type / Order Criteria	Absolute Orbit	XV_ORBIT_ABS	0
	Relative Orbit	XV_ORBIT_REL	1
zone_vis_time cover age outputs	Zone completely covered by swath	XV_COMPLETE	0
	Left extreme transition found by ZONE_VIS_TIME	XV_LEFT	1
	Right extreme transition found by ZONE_VIS_TIME	XV_RIGHT	2
	Both extreme transition found by ZONE_VIS_TIME	XV_BOTH	3
stat_vis_time mask inputs	AOS, LOS and physical masks	XV_COMBINE	0
	AOS, LOS masks	XV_AOS_LOS	1
	Physical mask only	XV_PHYSICAL	2
	Mask as from Station file	XV_FROM_FILE	3
star_vis_time cover age outputs	Visibility stars/ends at the first/last FOV in star_vis_time	XV_STAR_UNDEFINED	0
	Visibility stars/ends at the upper FOV in star_vis_time	XV_STAR_UPPER	1
	Visibility stars/ends at the lower FOV in star_vis_time	XV_STAR_LOWER	2
	Visibility stars/ends at the left FOV in star_vis_time	XV_STAR_LEFT	3
	Visibility stars/ends at the right FOV in star_vis_time	XV_STAR_RIGHT	4
Order enumeration	Input Segments ordered by start time	XV_TIME_ORDER	0
	Input Segments not ordered by start time	XV_NO_TIME_ORDER	1
Segments direction	Ascending segment	XV_ASCENDING	0
	Descending segment	XV_DESCENDING	1
Swath flag	Swath Template File	XV_STF	0
	Swath Definition File	XV_SDF	1
Swath id initialization	File automatic	XV_FILE_AUTO	0
	Swath definition file	XV_FILE_SDF	1
	Swath template file	XV_FILE_STF	2
	Swath definition data	XV_SDF_DATA	3

	Swath template data	XV_STF_DATA	4
Zone type	Use zone database	XV_USE_ZONE_DB_FILE	0
	Use zone data	XV_USE_ZONE_DATA	1
Extra information computation	Do not compute extra information	XV_DO_NOT_COMPUTE	0
	Compute extra information	XV_COMPUTE	1
Station type	Use station file	XV_USE_STATION_FILE	0
	Use station data	XV_USE_STATION_DATA	1
	Use station file and default mask	XV_USE_STATION_FILE_AND_MASK_OVERRIDE	2
	Use station data and default mask	XV_USE_STATION_DATA_AND_MASK_OVERRIDE	3
Segment time type	UTC time provided	XV_UTC_TYPE	0
	Orbit data provided	XV_ORBIT_TYPE	1
	UTC time and orbit data provided	XV_BOTH_TYPE	2
Coverage computation type	Coverage computation of point database done with a fixed distance between points	XV_COVERAGE_FIXED_DISTANCE	0
	Coverage computation performance aiming a given percentage of precision	XV_COVERAGE_PERCENTAGE_PRECISION	1

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 the current section is to present the data structures that are used in the EO_LIB library. The structures are currently used for the CFI Identifiers accessor functions. The following table show the structures with their names and the data that contain:

Table 4: EO_VISIBILITY structures

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_link_mask	incl_mask	xv_az_el_mask	List of azimuth and elevation pairs in Instrument Frame defining an inclusive zone
	excl_mask	xv_az_el_mask	List of azimuth and elevation pairs in Instrument Frame defining an exclusive zone
xv_link_data	mask_data	xv_link_mask	List of azimuth and elevation pairs in Instrument Frame
	min_tg_height	double	Minimum tangent height

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_swath_info	type	long	Initialization type (Swath initialization type enum)
	filename	char*	File name
	sdf_file	xd_sdf_file*	Swath definition data
	stf_file	xd_stf_file*	Swath template data
	nof_regen_orbits	long	Number of orbit for STF regeneration (for SDF initializations)

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_time	type	long	Time type (see Segment time type enum)
	utc_time	double	UTC time (processing days)
	orbit_type	long	Orbit type (see Orbit type enum)
	orbit_num	long	Orbit number
	cycle	long	Cycle number
	sec	long	Seconds since ascending node
	msec	long	Microseconds since ascending node

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_time_interval	tstart	xv_time	Interval start time
	tstop	xv_time	Interval stop time

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_zone_coverage_info	zone_id	char*	Zone name
	coverage	long	Zone coverage flag for segment = 0 Zone completely covered by swath = 1 Zone not completely covered by swath, extending over the left edge of the swath. = 2 Zone not completely covered by swath, extending over the right edge of the swath. = 3 Zone not completely covered by swath, extending over both edges of

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_zone_coverage_info_list	num_rec	long	Number of zones
	zone_coverage_info	xv_zone_coverage_info	List of zones with coverage
xv_zonevisibility_interval	time_interval	xv_time_interval	Segment start and stop
	zone_coverage_info_list	xv_zone_coverage_info_list	Coverage information
xv_zonevisibility_interval_list	num_rec	long	Number of segments
	visibility_interval	xv_zonevisibility_interval	List of segments

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_zone_info	zone_id	char*	Zone name
	type	type	Zone type (see Zone type enum)
	zone_db_filename	char*	Zone database name
	zone_data	xd_zone_rec*	Zone record information
	projection	long	Projection (see projection enum)
	min_duration	double	Minimum segment duration (seconds)

<table>
<tr><th rowspan="2">Structure name</th><th colspan="3">Data</th></tr>
<tr><th>Variable Name</th><th>C type</th><th>Description</th></tr>
<tr><td rowspan="4">xv_az_el_mask</td><td>num_mask_pt</td><td>long</td><td>Number of azimuth and elevation pairs defining the antenna mask</td></tr>
<tr><td>status</td><td>long</td><td>Allow the user to enable/disable masks;
The behaviour of the status field is described below for each type of mask:
Inclusive mask:
Status = disabled: no constraints (regardless of number of points)
Status = enabled and number of points = 0 : no constraints
Exclusive mask:
Status = disabled : mask is ignored (regardless of number of points)
Status = enabled and number of points = 0 : mask is ignored
Combining the two above:
Each mask define a polygon.
Forbidden areas are:
1) the area OUTSIDE the inclusive polygon;
2) the area INSIDE the exclusive polygon;</td></tr>
<tr><td>azimuth</td><td>double [XD_VERTICES]</td><td>Azimuth defining the antenna mask</td></tr>
<tr><td>elevation</td><td>double [XD_VERTICES]</td><td>Elevation defining the antenna mask</td></tr>
<tr><td rowspan="3">xv_zone_info_list</td><td>calc_flag</td><td>long</td><td>Flag to tell if extra information must be computed (see extra information compute flag enum)</td></tr>
<tr><td>num_rec</td><td>long</td><td>Number of zones</td></tr>
<tr><td>zone_info</td><td>xv_zone_info*</td><td>List of zones.</td></tr>
</table>

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_station_info	type	long	See station type enum.
	station_id	char*	Station name
	station_db_filename	char*	Station database
	station_data	xd_station_rec*	Station data
	default_aos	double	Default AOS elevation
	default_los	double	Default LOS elevation
	default_mask	long	Default mask

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_station_info_list	calc_flag	long	Flag to tell if extra information must be computed (see extra information compute flag enum)
	num_rec	long	Number of stations
	station_info	xv_station_info*	List of stations.
xv_station_coverage_info	station_id	char*	Name of the station
	zdop_time	xv_time	Zero doppler time

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_station_coverage_info_list	num_rec	long	Number of stations with coverage information
	station_coverage_info	xv_station_coverage_info*	List of stations
xv_stationvisibility_interval	time_interval	xv_time_interval	Segment start/stop information
	station_coverage_info_list	xv_station_coverage_info_list	Segment extra information
xv_stationvisibility_interval_list	num_rec	long	Number of segments
	visibility_interval	xv_stationvisibility_interval	List of segments

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_swath_point	lon	double	Longitude
	lat	double	Latitude
	alt	double	Altitude
xv_swath_point_list	num_rec	long	Number of points
	swath_point	xv_swath_point*	List of swath points
xv_visibility_interval	time_interval	xv_time_interval	Time interval

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_visibility_interval_list	num_rec	long	Number of segments
	visibility_interval	xv_visibility_interval*	List of segments

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_zonevisibility_coverage_in	type_coverage	long	Coverage type (see XV_Type_Coverage_enum)
	point_geod_distance	double	Geodetic distance between points [km]
	percent_precision	double	Expected precision. The closer to 100%, the more accurate the computations
	orbit_id	xo_orbit_id*	Orbit id
	attitude_def	xp_attitude_def*	Attitude definition
	swath_id	xv_swath_id*	Swath id

Structure name	Data		
	Variable Name	**C type**	**Description**
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	status	long	Allow the user to enable/disable masks; The behaviour of the status field is described below for each type of mask: **Inclusive mask:** Status = disabled: no constraints (regardless of number of points) Status = enabled and number of points = 0 : no constraints **Exclusive mask:** Status = disabled : mask is ignored (regardless of number of points) Status = enabled and number of points = 0 : mask is ignored **Combining the two above:** Each mask define a polygon. Forbidden areas are: 1) the area OUTSIDE the inclusive polygon; 2) the area INSIDE the exclusive polygon;
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_zonevisibility_coverage_out	zone_area	double	Zone area in km^2
	total_coverage	double	Percentage of zone covered by all intervals, i.e. 100% - percentage of zone not covered by any interval
	coverage_per_interval	double*	Array, item with index i (0,1,2,...N-1) is the percentage of zone covered by interval i+1 (1,2,...N) only
	coverage_by_N_intervals	double*	Array, item with index i (0,1,2,...N-1) is the percentage of the zone covered by exactly i+1 (1,2,...N) intervals
	cumulative_coverage	double*	Array, with index i (0,1,2,...N-1) is the percentage of zone covered by

7 CFI FUNCTIONS DESCRIPTION

The following sections describe each CFI function.

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])

7.1 xv_zone_vis_time

7.1.1 Overview

Note: this function is deprecated. Use **xv_zonevistime_compute** instead.

The **xv_zone_vis_time** function computes all the orbital segments for which a given instrument swath intercepts a user-defined zone at the surface of the Earth ellipsoid.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as seconds (and microseconds) elapsed since the ascending node crossing.

A user-defined zone can be:

- a polygon specified by a set of latitude and longitude points
- a circle specified by the centre latitude, longitude, and the diameter

Note that particular cases of the above can be used to define the zone as:

- a point
- a line

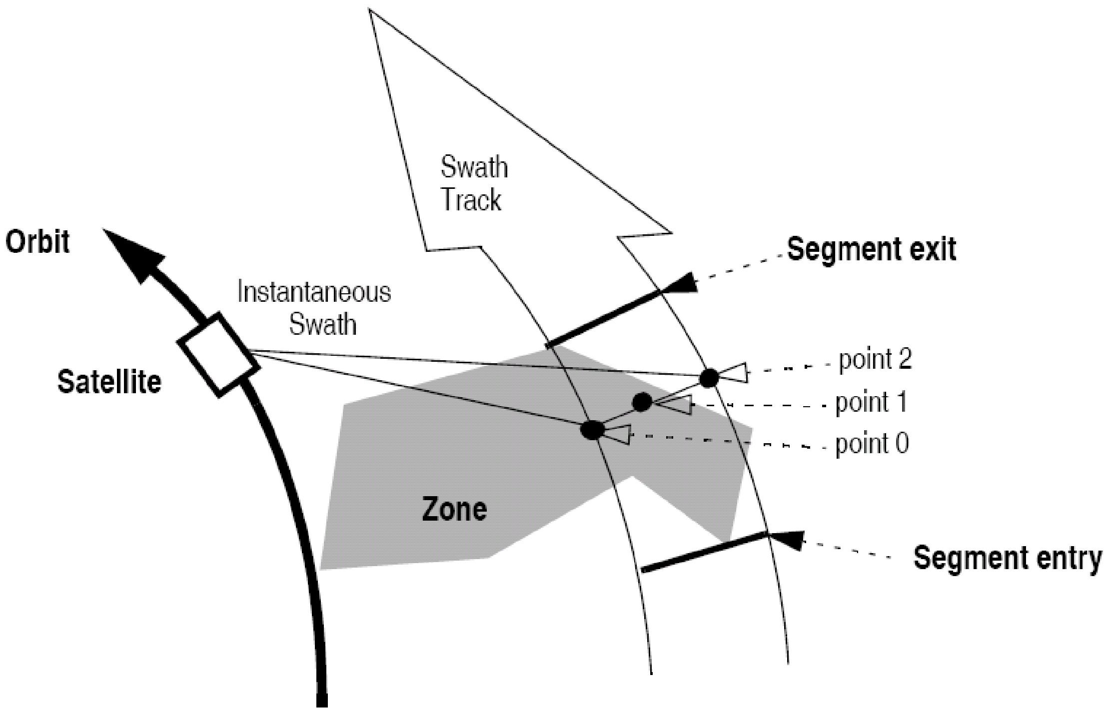


Figure 2: Segment Definition xv_zone_vis_time

xv_zone_vis_time requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files, also with precise propagator initialization if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files

  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**).
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
  - TLE files

    Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.
- the Instrument Swath File, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit. The Swath data can be provided by:
  - A swath template file produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).
  - A swath definition file, describing the swath geometry. In this case the **xv_zone_vis_time** generates the swath points for a number of orbits given by the user.
- optionally, a Zone Database File, containing the zone description. The user can either specify a zone identifier referring to a zone in the file, or provide the zone parameters directly to **xv_zone_vis_time.**

The time intervals used by **xv_zone_vis_time** are expressed in absolute orbit numbers or in relative orbit and cycle numbers. This is valid for both:

- input parameter "Orbit Range": first and last orbit to be considered. In case of using relative orbits, the corresponding cycle number should be used, otherwise, this the cycle number will be a dummy parameter.
- output parameter "Zone Visibility Segments": time segments with time expressed as {absolute orbit number (or relative orbit number and cycle number), number of seconds since ascending node, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

Users who need to use processing times must make use of the conversion routines provided in EO_ORBIT (**xo_time_to_orbit** and **xo_orbit_to_time** functions).

NOTE: If **xv_zone_vis_time** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behaviour depends on the swath file introduced as input:

•If a **swath template file** is used, **xv_zone_vis_time** automatically will ignore the orbits that do not correspond with the template file (i.e. no visibility segments will be generated for those orbits), since swath template file is generated from a reference orbit with a particular geometry, so it is not valid for a different geometry.

•If a **swath definition file** is introduced, **xv_zone_vis_time** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath template files valid for every orbital change are generated to perform the calculations.

NOTE 2:If a swath template file with the variable header tags *Start_Validity_Range* and *Stop_Validity_Range* is used as input, only the segments belonging to that orbit range will be returned.

NOTE 3: If a swath definition file is introduced, it can be also introduced every how many orbits the swath template file must be recomputed (swath_flag parameter, see section 68). If the orbit_id has been initialized with an OSF file with MLST non linear terms and the parameter swath_flag is greater than the linear approximation validity, the recomputation of swath template file will be done every linear approximation validity orbits.

7.1.2 Swath Definition

The swath file is generated using the xv_gen_swath function, within the EO_VISIBILITY library. There are 3 different types of swaths:

- earth-observing instruments (‘nadir curve’, ‘nadir point’ or “area swaths”)
- limb-sounding instruments (‘limb’, narrow or wide)
- limb-sounding instruments observing inertial objects (‘inertial’)

The following sub-sections provide some details on the various swath definitions.

7.1.2.1 Earth-observing Instruments Swath Definition

The term swath must be clearly defined to understand the explanations in this document:

- instantaneous swath: the part of the earth surface observed by an instrument at a given time
- swath track: represents the track made on the earth surface by the instantaneous swath over a period of time

For instruments observing the surface of the earth, the instantaneous swath is constituted by the point/ curve/area on the ground observed by the instrument at a given time. It is calculated taking the earth ellipsoid as a reference for the earth surface. The wider the field-of-view of the instrument, the wider the swath on the ground.

When the satellite moves over a period of time, this point/curve/area defines a band on the earth surface. This constitutes the swath track.

See next figure for an illustration of these definitions.

Note that the terms curve or point are an idealized view of the instrument FOV, which usually have a thickness.

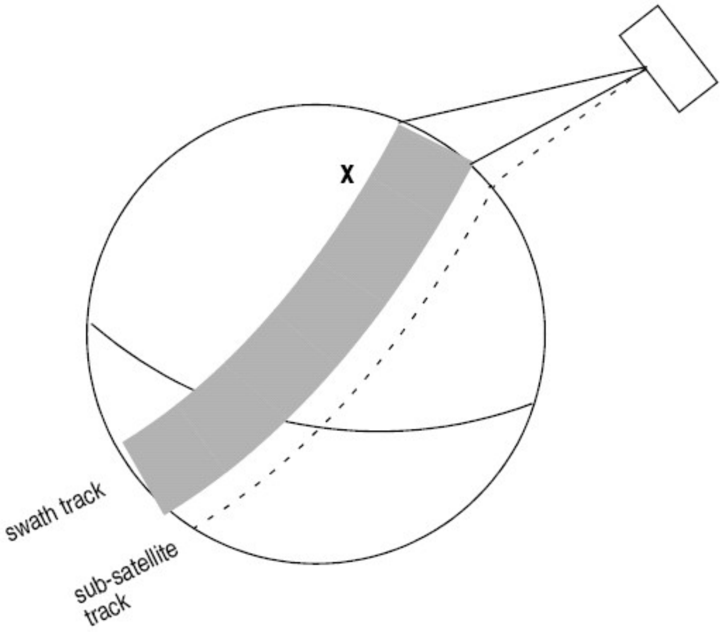


Figure 3: Earth-observing instrument: swath definition

7.1.2.2 Limb-sounding Instruments Swath Definition

For limb sounding instruments, the concept can be generalized to define a “thick swath”. This is obtained by defining a minimum and a maximum altitude, and considering the tangent points to these altitudes as the edges of the swath. Two cases have to be considered:

- deterministic (narrow) azimuth field of view (e.g. MIPAS sideward-looking): the swath projection on the earth surface is similar to a regular sideward-looking swath, with the lower altitude defining the further swath edge and the higher altitude defining the closer swath edge. See Figure 4.
- non-deterministic (potentially wide) azimuth field of view (e.g. MIPAS rearward-looking): due to the potentially wide azimuth field of view, each altitude defines a swath projection on the earth surface. Depending on the altitude, these swaths are of different width across-track, and also at different distance from the satellite. See Figure 5.

For these, 2 Instrument Swath Files are provided:

- one at the highest altitude
- one at the lowest altitude

The user must handle both swath himself to determine his required visibility time segments

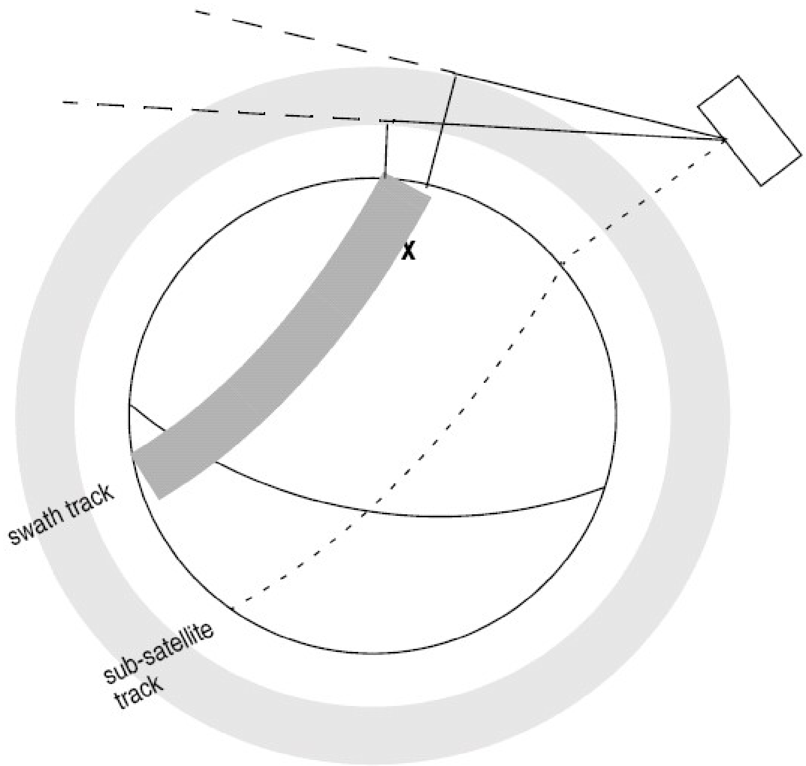


Figure 4: Limb-sounding instrument: swath definition (1)

.

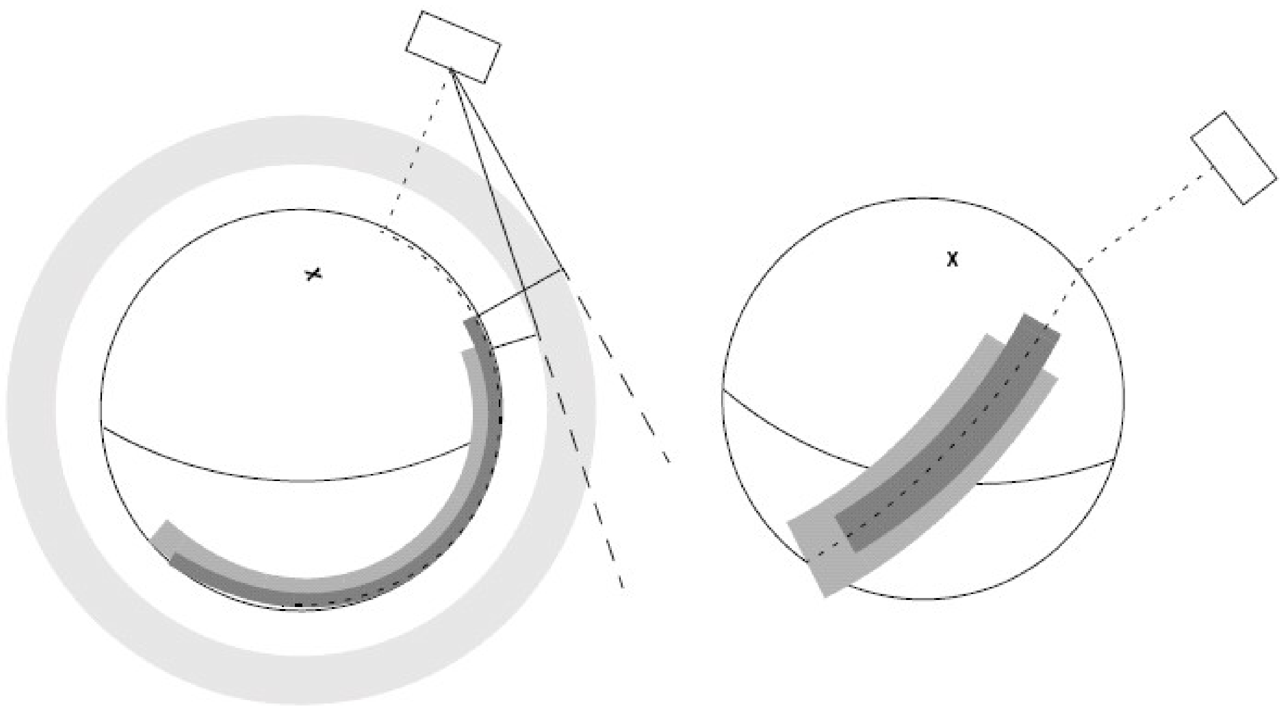


Figure 5: Limb-sounding instrument: swath definition (2)

7.1.2.3 Limb-sounding Instruments Inertial Swath Definition

This type corresponds to the observation of inertial targets (e.g. Gomos occultation mode and Mipas Line of Sight mode in Envisat). For the CFI function **xv_star_vis_time** the FOV direction in inertial coordinates must be available. Therefore for these instrument modes the direction in inertial space, for a given tangent altitude, is given in the swath template file.

7.1.2.4 Swath Definition for Envisat

Next table lists all instrument modes and the relevance of the swaths for Envisat. It shows also:

- the prefix to be used when generating the swath template file name
- the different types of algorithms to be used by xv_gen_swath (this is transparent to the user)

Table 5: Envisat Swaths

instrument	Mode	File Prefix = swath	Swath geometry (Table 90)	Swath Type	Remarks
RA		RA_2__	POINTING (1 point)	Nadir point	Modeled as sub-satellite track
MERIS	Averaging / Direct & Averaging	MERIS_	POINTING (3 points)	Nadir line	
ASAR	Image Modes (IS1... IS7)	SARxIM (x=1...7)	ASAR	Nadir line	
	Alt. Polarization (IS1... IS7)				
	Wide Swath	SARWIM			
	Global Monitoring				
	Wave (IS1... IS7)	SARxWV (x=1...7)			Modeled as a continuous swath any where within the image swath

GOMOS	Occultation	GOMOIL GOMOIH	INERTIAL	Inertial direction	IFOV much smaller than swath. IFOV Very dependent on star avail ability. 2 swaths defined: - 1 for high altitude (GOMOIH) - 1 for low altitude (GOMOIL)
	Occultation	GOMO_H GOMO_L	LIMB	Limb wide	Same mode as above, now swath defined as Earth-fixed location. IFOV much smaller than swath. IFOV Very dependent on star avail ability. 2 swaths defined: - 1 for high altitude (GOMO_H) - 1 for low altitude (GOMO_L)
SCIAMACHY	Nadir / Nadir of Nadir & Limb	SCIAN_	POINTING (3 points)	Nadir line	Continuous Nadir swath modeled
	Limb / Limb of Nadir & Limb	SCIALH SCIALL		Limb wide	2 swaths defined: - 1 for high altitude (SCIALH) - 1 for low altitude (SCIALL)
AATSR		ATSR_N ATSR_F	POINTING (3 points)	Nadir line	2 swaths defined: - 1 for nadir swath - 1 for forward swath
MWR		MWR___	POINTING (1 points)	Nadir point	Modeled as sub-satellite track
MIPAS	Nominal	MIPN_H MIPN_L	LIMB	Limb narrow	2 swaths defined: - 1 for high altitude (MIPN_H) - 1 for low altitude (MIPN_L)
	Special Event Mode (across)	MIP_X_	LIMB	Limb narrow	Modeled as an across track swath, in the middle of the MIPAS SEM acquisition scan.
	Special Event Mode (rearward)	MIP_RH MIP_RL	LIMB	Limb wide	IFOV much smaller than swath. 2 swaths defined: - 1 for high altitude (MIP_RH) - 1 for low altitude (MIP_RL)
	Rearward Sideward	MIPIRH MIPIRL MIPIXH	INERTIAL	Inertial direction	2 swaths defined for rearward mode: - 1 for high altitude (MIPIRH) - 1 for low altitude (MIPIRL)

		MIPIXL			3 swaths defined for sideward mode: - 1 for high altitude (MIPIXH) - 1 for back mode (MIPIXB) - 1 for forward mode (MIPIXF)

7.1.3 Zone Borders and Projection

When defining a polygon zone, the user is assumed to wish polygon sides as straight lines. But on the earth surface, a straight line is, at best, a confusing concept.

The only way to define unambiguously straight lines is to work in a 2-dimensional projection of the earth surface. There are many possible projections, each having advantages and drawbacks.

xv_zone_vis_time can handle zone borders in 2 different projections:

- rectangular projection, using longitude and latitude as the X and Y axis; this is appropriate to express zones where (some of) the edges follow constant latitude lines, and provide a reasonable approximation for straight lines at low-medium latitudes
- azimuthal gnomonic projection, where great circles are always projected as straight lines; this is better for high latitudes, where the rectangular projection suffers from too much distortion and the singularity at the poles.

xv_zone_vis_time allows the user to specify which projection he wants to work in, i.e. in which projection the polygon sides will be represented by **xv_zone_vis_time** as straight lines. The user is assumed to be aware of how the polygon sides behave on the Earth surface.

7.1.4 Zone Definition

The user-defined zone can be either (see Table 6);

- a point
- a line
- a polygon
- a circle

A zone is defined by the area of the earth surface enclosed by the zone borders:

- in the case of a circular zone, the area inside the circle
- in the case of a polygonal zone, the area which is always to the right of any polygon side; if the polygon is defined as a sequence of N points, each polygon side is considered as a line from point i to point i+1; this unambiguously defines the right side of the polygon sides.

Table 6: Zone definition

Zone definition	Zone_num	Zone_long Zone_lat	Zone_diam	Description
Circular Zone	1	[0]: centre point	yes zone_diam > 0.0	The zone is represented as a circle, around the centre point
Point Zone	1	[0]: Point	yes zone_diam = 0.0	The zone is defined by the point. Resulting segments will have a zero duration. The zone will always be completely covered by the swath.
Line Zone	2	[0], [1]: Line	no	The zone is defined by the line from point [0] to point [1].
Polygon Zone	>2	[i]	no	The zone is defined by the area right of the line from point [i] to point [i+1].

For the gnomonic projection, a side of a zone is always smaller than a half great circle, because two polygon points are considered to be joined by the shortest line.

For the rectangular projection, two consecutive points of the zone are also joined by the shortest line; so the difference in longitude must be less than 180 degrees.

The polygon zone can be closed (i.e. the first and last points are the same) or not. If the zone is not closed, **xv_zone_vis_time** closes it by joining the last point with the first one in its internal computations.

See Figure 6 for examples of zone definitions.

xv_zone_vis_time will issue an error on the zone definition if the polygon has intersecting sides ("butterfly" zone).

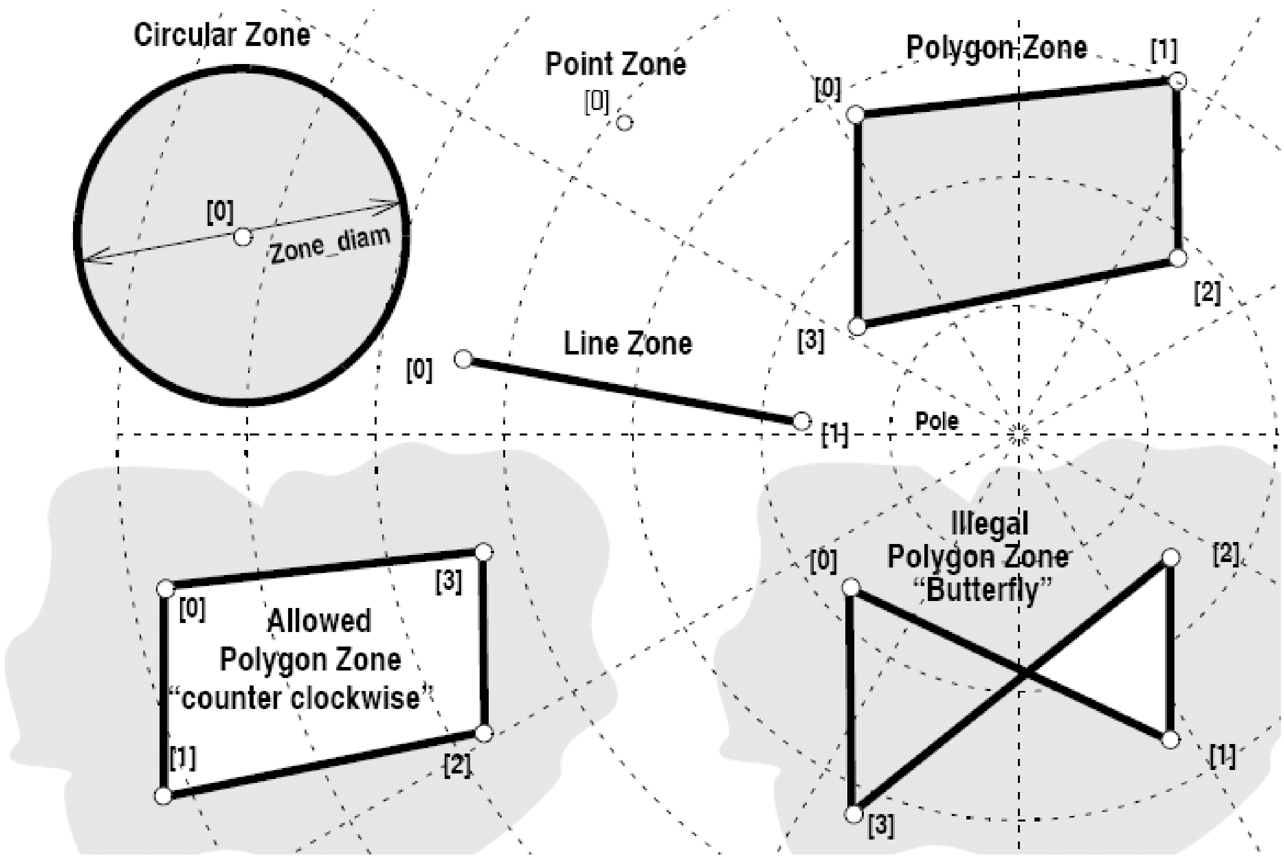


Figure 6: Zone examples

7.1.5 Intersection Definition

The **xv_zone_vis_time** intersection times between the instrument swath and the user-defined zone are defined as the first and last occurrence, in chronological order with respect to the satellite direction, of the geometrical super-position of any point belonging to the instrument swath with any single point belonging to the zone (including the zone border).

The entry and exit times for each intersection are given as elapsed seconds (and microseconds) since the ascending node crossing.

Next figure shows some typical intersections.

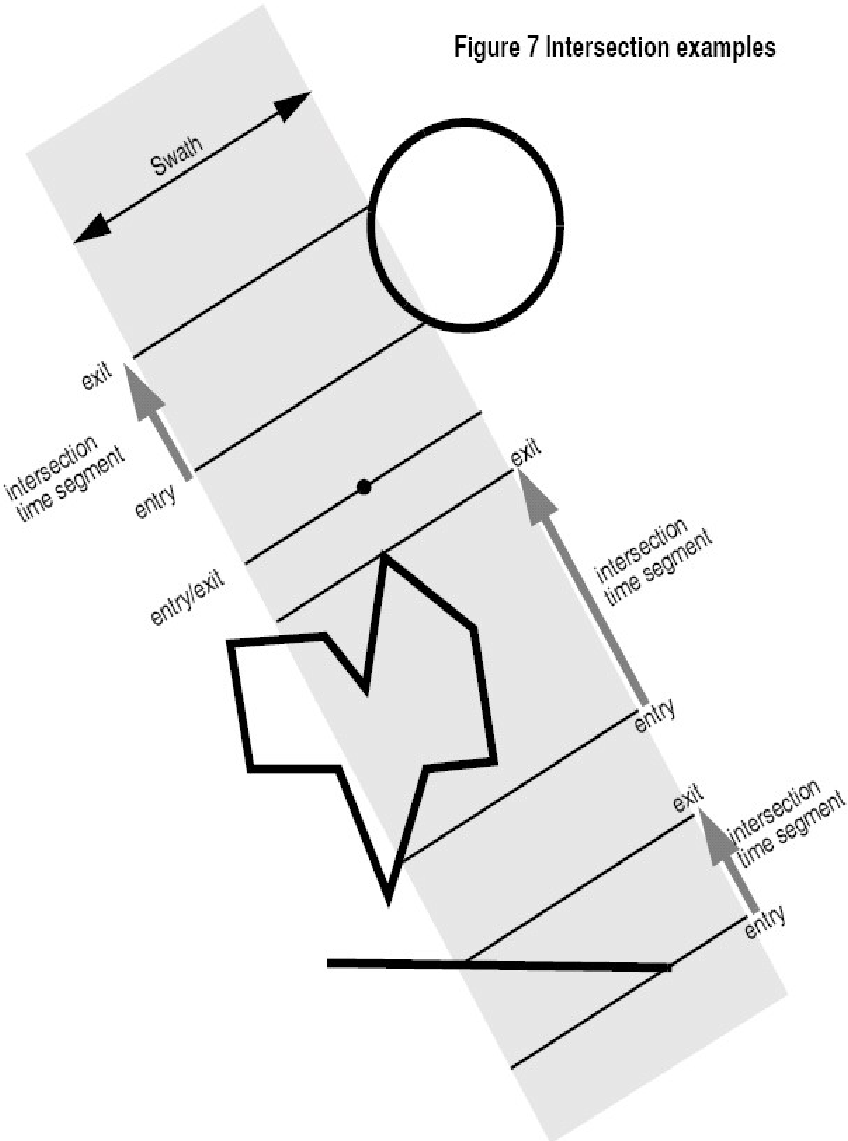


Figure 7: Intersection examples

7.1.6 Intersection Algorithm

The intersection of a swath and a user-defined zone is to be performed on the Earth projected to a map plane in one of the following projections:

- Rectangular projection
- Gnomonic projection

Although the projections are quite different, the intersection rules are identical. The algorithm can however be different, in order to take advantage of a particular feature of a projection.

The purpose of the CFI function ZONEVISTIME is to obtain quickly, accurate intersection segments with a low precision (1 second).

The algorithms assume that the polygon zones are closed and expects a wrap around between the first and the last point. Thus ZONEVISTIME must first close the polygon if necessary.

For ZONEVISTIME the following swath types are defined:

- point swath: instantaneous swath is a point.
- segment swath: instantaneous swath is a segment.
- multi-segment swath: it can be open or closed.
- inertial swath: not used by ZONEVISTIME

The main concept in the algorithm is the transition, defined as the change in coverage of (part of) the swath and the zone (e.g. edge of the swath crosses one polygon side).

7.1.6.1 Intersection with a point swath

The vertices of the polygon defining the area are connected by straight lines in the chosen projection, along track swath points are also connected by straight lines in the same projection.

Transitions are located by linear intersection of the zone sides and the swath along track lines. A transition is only valid if the intersection occurs inside both line segments. The polygon side from <i> to <j> is defined in a clockwise manner inclusive point <i> but exclusive point <j>. The swath line from time <k> to <l> is defined inclusive the template point at <k> but exclusive the template point at <l>.

The fraction of the swath along track line determines the precise timing since time <k> of the intersection. Also the determination if the transition is a on- or off-transition is quite trivial. First a vector is defined, perpendicular to the along track swath line, such that the vector points left. Then, the dot product of the polygon side and this vector is calculated. If the dot product is positive, the transition is on, i.e. the swath enters the zone. If the result is negative, then the swath leaves the zone. If the result equals zero then the transition can be ignored (polygon side and swath overlay, a proper transition will be found with another pair of polygon side - swath line.).

7.1.6.2 Intersection with a segment swath

The left and right side of the swath, are located using the same algorithm as for the point swath. Even left and right time segments can be made based on the left and right hand transitions.

The polygon vertices (and not the sides) are intersected with the along track moving line swath, in order to catch zones smaller than the swath, etc. Swaths for intermediate times between two consecutive times in Swath Template File are considered straight segments, joining an intermediate point of the Left swath line from time <k> to time <l>, with an intermediate point in Right swath line.

7.1.6.3 Intersection with a multi-segment swath

The algorithm used for segment swath is repeated for every segment of the swath, and the visibility segments obtained in each case are merged with the ones of the other swath segments.

For a closed swath further calculations are done: it is checked if the zone is completely inside the swath area in the interval between contiguous visibility segments, or between the beginning of the first orbit and the first visibility segment, or between the last visibility segment and the end of the last orbit computed. If it is inside, segments must be merged because the zone was visible in the interval.

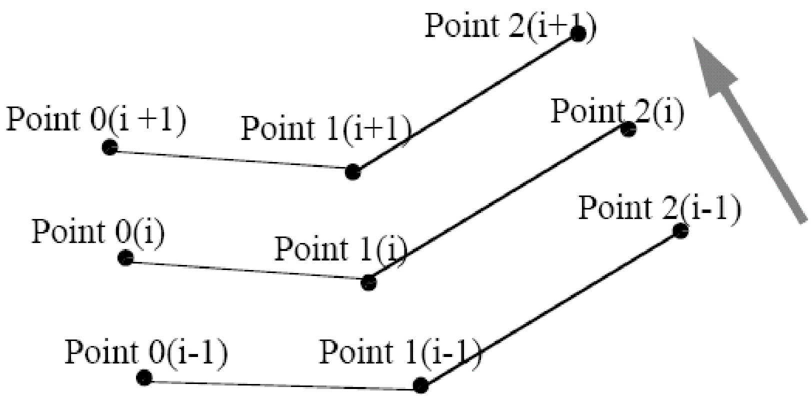


Figure 8: Swath points

7.1.7 Usage Hints

7.1.7.1 Limb-sounding Instruments Intersection

In the case of limb-sounding instrument with a potentially wide azimuth field of view, 2 swaths have to be considered (1 for minimum altitude, 1 for maximum altitude). Furthermore, these 2 swaths are offset in time (i.e. their projection on the earth intersect with a given point at different times). To cope with this, the user must do the following:

- call **xv_zone_vis_time** twice (once for each extreme altitude swath)
- merge/filter the 2 sets of time segments, depending on what he wants to achieve

7.1.7.2 Zone Coverage

xv_zone_vis_time computes purely geometrical intersections. The resulting zone visibility segments might need some additional filtering by the user. In particular, instrument constraints (e.g. only working outside of sun eclipse) have to be considered by the user.

Furthermore, to help users to deal with zones wider than the swath (i.e. requiring several orbits to cover the whole zone), **xv_zone_vis_time** produces for each zone visibility segment an indication of the coverage type (see Figure 9);

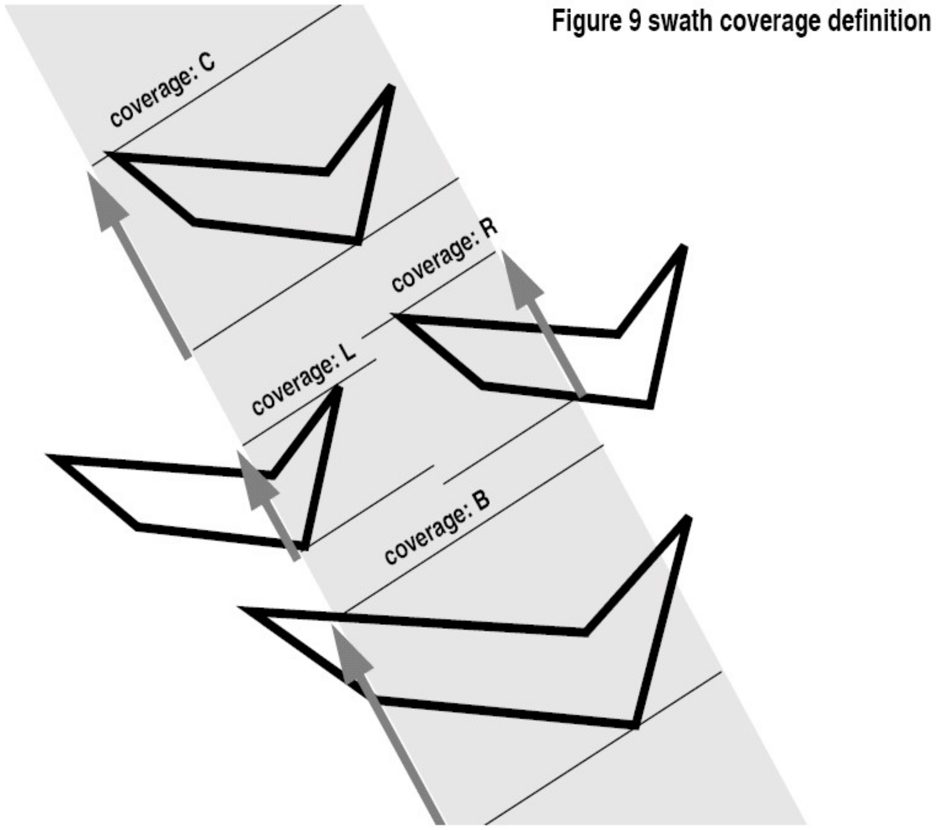


Figure 9: Swath coverage definition

- coverage = C: zone completely covered by the swath
- coverage = R: zone partially covered by the swath, extending over the right edge of the swath
- coverage = L: zone partially covered by the swath, extending over the left edge of the swath
- coverage = B: zone partially covered by the swath, extending over both edges of the swath

7.1.7.3 Combined use of xv_swath_pos and the coverage flag

The EO_VISIBILITY function xv_swath_pos can be used to refine the work performed with **xv_zone_vis_time**.

7.1.8 Calling sequence

For C programs, the call to **xv_zone_vis_time** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xo_orbit_id orbit_id = {NULL};
       long      swath_flag, orbit_type,
                 start_orbit, start_cycle,
                 stop_orbit, stop_cycle,
                 zone_num, projection,
                 number_segments,
                 *bgn_orbit, *bgn_second,
                 *bgn_microsec, *bgn_cycle,
                 *end_orbit, *end_second,
                 *end_microsec, *end_cycle,
                 *coverage, ierr[XV_NUM_ERR_ZONE_VIS_TIME],
                 status;
       double    *zone_long, *zone_lat,
                 zone_diam, min_duration;
       char      *swath_file;
       char      *zone_id, *zone_db_file;

       status = xv_zone_vis_time(&orbit_id,
                             &orbit_type,
                             &start_orbit, &start_cycle,
                             &stop_orbit, &stop_cycle,
                             &swath_flag, swath_file,
                             zone_id, zone_db_file,
                             &projection, &zone_num,
                             zone_long, zone_lat, &zone_diam,
                             &min_duration,
                             &number_segments,
                             &bgn_orbit, &bgn_second,
                             &bgn_microsec, &bgn_cycle,
                             &end_orbit, &end_second,
                             &end_microsec, &end_cycle,
                             &coverage, ierr);
```

```
        /* Or, using the run_id */
        long run_id;

        status = xv_zone_vis_time_run(&run_id,
                                      &orbit_type,
                                      &start_orbit, &start_cycle,
                                      &stop_orbit, &stop_cycle,
                                      &swath_flag, swath_file,
                                      zone_id, zone_db_file,
                                      &projection, &zone_num,
                                      zone_long, zone_lat, &zone_diam,
                                      &min_duration,
                                      &number_segments,
                                      &bgn_orbit, &bgn_second,
                                      &bgn_microsec, &bgn_cycle,
                                      &end_orbit, &end_second,
                                      &end_microsec, &end_cycle,
                                      &coverage, ierr);
}
```

7.1.9 Input parameters

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

Table 7: Input parameters of xv_zone_vis_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long*	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters. Relative orbits only can be used when the orbit_id was initialized with orbital changes (with xo_orbit_init_def or with xo_orbit_init_file plus an OSF file). In other cases, only the value XV_ORBIT_ABS can be used.	-	Complete.
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit. First Orbit for the orbit initialization will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤repeat cycle
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using absolute orbits	cycle number	= 0 or ≥ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. For orbit_id initialized with orbital changes, when: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) or • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤repeat cycle

			change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)		
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using absolute orbits	cycle number	= 0 or ≥ first cycle in osf
swath_flag	long*	-	Define the use of the swath file: • 0 = (XV_STF) if the swath file is a swath template file. • > 0 if the swath files is a swath definition file. In this case the swath points are generated for every “swath_flag” orbits	-	XV_STF = 0 XV_SDF = 1 > 0
swath_file	char *	-	File name of the swath-file for the appropriate instrument mode		
zone_id	char*		Identification of the zone, as defined in zone_db_file. This parameter is used ONLY IF zone_num = 0		
zone_db_file	char *		File name of the zone-database-file. This file is used ONLY IF zone_num = 0		
projection	long		projection used to define polygon sides as straight lines: = 0 Read projection from Zones DB = 1 Azimuthal gnomonic = 2 Rectangular lat/long		
zone_num	long		Number of vertices of the zone provided in zone_long, zone_lat: = 0 no vertices provided, use zone_id / zone_db_file = 1 Point / Circular zone, = 2 Line zone > 2 Polygon zone		≥ 0
zone_long	double*	all	zone_long[i-1] Geocentric longitude of - circle centre, for circ. zone, i =1 - point, for point zone, i = 1 - line-end, for line zone, i = 1 or 2		

			- vertices, for polygon zone, i = 1... zone_num		
zone_lat	double*	all	zone_lat[i-1] Geodetic latitude of - circle centre, for circ. zone, i =1 - point, for point zone, i = 1 - line-end, for line zone, i = 1 or 2 - vertices, for polygon zone, i = 1... zone_num		
zone_diam	double		Zone diameter for circular zones, dummy for other zones If diameter equals 0.0 then zone is Point Zone	m	≥ 0.0
min_duration	double		Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0

It is also possible to use enumeration values rather than integer values for some of the input arguments, as shown in the table below:

Input	Description	Enumeration value	long
projection (defined in [D_H_SUM])	Read projection from the zones DB file	XD_READ_DB	0
	Azimuthal Gnomonic	XD_GNOMONIC	1
	Rectangular long/lat	XD_RECTANGULAR	2

7.1.10 Output parameters

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

Table 8: Output parameters of xv_zone_vis_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xv_zone_vis_time	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long		Number of visibility segments returned to the user.		≥ 0
bgn_orbit	long*	all	Orbit number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		> 0
bgn_second	long*	all	Seconds since ascending node, begin of visibility segment i bgn_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period
bgn_microsec	long*	all	Micro seconds within second begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments	µs	≥0 999999
bgn_cycle	long*	all	Cycle number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
end_orbit	long*	all	Orbit number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		> 0
end_second	long*	all	Seconds since ascending node, end of visibility segment i end_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period

end_microsec	long*	all	Micro seconds within second end of visibility segment i end_microsec[i-1], i = 1, number_segments	µs	≥0 999999
end_cycle	long*	all	Cycle number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
coverage	long*	all	Zone coverage flag for segment = 0 Zone completely covered by swath = 1 Zone not completely covered by swath, extending over the left edge of the swath. = 2 Zone not completely covered by swath, extending over the right edge of the swath. = 3 Zone not completely covered by swath, extending over both edges of the swath coverage[i], i = 0, (number_segments-1)		
ierr[XV_NUM_ERR_ZONE_VIS_TIME]	long		Error status flags		

It is also possible to use enumeration values rather than integer values for some of the output arguments, as shown in the table below:

Input	Description	Enumeration value	long
coverage	Zone completely covered by swath	XV_COMPLETE	0
	Left extreme transitions found	XV_LEFT	1
	Right extreme transitions found	XV_RIGHT	2
	Both extreme transitions found	XV_BOTH	3

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_zone_vis_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.1.11 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_zone_vis_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_zone_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code.**

Table 9: Error messages and codes for xv_zone_vis_time

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Input parameter "Number of ZONE points" is wrong.	Computation not performed	XV_CFI_ZONE_VIS_TIME_NEGATIVE_NUM_ZONE_ERR	0
ERR	Input parameter "Orbit Id" is wrong.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_STATUS_ ERR	1
ERR	Input parameter "orbit_type" is out of range.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_TYPE_ER R	2
ERR	Input parameter "Minimum duration" cannot be negative.	Computation not performed	XV_CFI_ZONE_VIS_TIME_NEGATIVE_MIN_DURATION_ERR	3
ERR	Input parameter "Projection" out of range.	Computation not performed	XV_CFI_ZONE_VIS_TIME_PROJECTION_OUT_OF_RANGE_ERR	4
ERR	Wrong swath_flag value	Computation not performed	XV_CFI_ZONE_VIS_TIME_SWATH_FLAG_E RR	5
ERR	Swath file is not compatible with the orbit file	Computation not performed	XV_CFI_ZONE_VIS_TIME_WRONG_SWATH_ERR	6
ERR	Could not generate the swath template file	Computation not performed	XV_CFI_ZONE_VIS_TIME_GENSWATH_ERR	7
ERR	Error generating visibility segments for orbit "%d"		XV_ZONE_VIS_TIME_IN_ORBIT_ERR	8
ERR	Error reading Swath Template File.	Computation not performed	XV_CFI_ZONE_VIS_TIME_READ_SWATH_FILE_ERR	9
ERR	Swath type not allowed	Computation not performed	XV_CFI_ZONE_VIS_TIME_INCORRECT_SWATH_TYPE_ERR	10
ERR	Cannot allocate memory for	Computation not performed	XV_CFI_ZONE_VIS_TI	11

	the Swath Template File		ME_ALLOCATE_SWATH_MEMORY_ERR	
ERR	Input parameter "start_orbit" cannot be negative.	Computation not performed	XV_CFI_ZONE_VIS_TIME_NEGATIVE_START_ORBIT_ERR	12
ERR	Error reading OEF/OSF file.	Computation not performed	XV_CFI_ZONE_VIS_TIME_READ_OSF_ERR	13
WAR N	"start_orbit" is before the first orbit in "orbit_event_file".	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_EARLIER_START_ORBIT_WARN	14
WAR N	"stop_orbit" is after the last orbit in "orbit_event_file".	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_LATER_STOP_ORBIT_WARN	15
ERR	Input parameter "start_orbit" cannot be greater than "stop_orbit".	Computation not performed	XV_CFI_ZONE_VIS_TIME_WRONG_ORBIT_RANGE_ERR	16
ERR	Error calling "xv_orbitinfo".	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBITINFO_CALL_ERR	17
ERR	"cycle_length" read from the input "Swath Template File" is not equal to that of any orbits within the orbit range	Computation not performed	XV_CFI_ZONE_VIS_TIME_INCONSISTENT_SWATH_ERR	18
WAR N	There is at least one orbital change within the requested orbit range.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_ORBITAL_CHANGE_WARN	19
ERR	Input parameter "zone_id" is an empty string.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ZONE_ID_EMPTY_ERR	20
ERR	Number of characters in input string "zone_id" is different from %li.	Computation not performed	XV_CFI_ZONE_VIS_TIME_WRONG_ZONE_ID_LENGTH_ERR	21
ERR	Error reading the ZONE Database file.	Computation not performed	XV_CFI_ZONE_VIS_TIME_READ_ZONE_DB_FILE_ERR	22
WAR N	"Projection" parameter set to default.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_DEFAULT_PROJECTION_WARN	23
ERR	Cannot allocate memory for the ZONE records."	Computation not performed	XV_CFI_ZONE_VIS_TIME_ALLOCATE_ZONE_MEMORY_ERR	24
ERR	Latitude must be in the range [-90.0 , 90.0].	Computation not performed	XV_CFI_ZONE_VIS_TIME_WRONG_LATITU	25

			DE_RANGE_ERR	
WAR N	Two consecutive points are equal, only one is used.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_TWO_EQUAL_POINTS_WARN	26
ERR	Difference in longitude for 2 consecutive ZONE points is equal to 180.0 degrees (RECTANGULAR projection). Zone definition is ambiguous.	Computation not performed	XV_CFI_ZONE_VIS_TIME_DIFF_LONG_180_ERR	27
ERR	Two consecutive ZONE points are antipodal (GNOMONIC projection). Zone definition is ambiguous.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ANTIPODAL_POINTS_ERR	28
ERR	Error precomputing intersection of two segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SEGMENT_INTERSECT_PREC_ERR	32
ERR	Error computing intersection of two segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SEGMENT_INTERSECT_COMP_ERR	33
ERR	Error computing gnomonic coordinates.	Computation not performed	XV_CFI_ZONE_VIS_TIME_GNOMONIC_COORD_ERR	34
ERR	Two ZONE segments intersect.	Computation not performed	XV_CFI_ZONE_VIS_TIME_TWO_SEGMENTS_INTERSECT_ERR	35
ERR	Two consecutive ZONE segments are aligned in the same direction.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ALLIGNED_SEGMENTS_ERR	36
ERR	Input parameter "ZONE diameter" cannot be negative (POINT or CIRCLE zone).	Computation not performed	XV_CFI_ZONE_VIS_TIME_ZONE_DIAM_NEGATIVE_ERR	37
ERR	SWATH contains the POLE (RECTANGULAR projection).	Computation not performed	XV_CFI_ZONE_VIS_TIME_POLE_IN_SWATH_ERR	38
ERR	Not convex SWATH quadrilateral for the specified latitude range.	Computation not performed	XV_CFI_ZONE_VIS_TIME_CUADRILATERAL_NOT_CONVEX_ERR	39
ERR	Error checking if a point is inside a quadrilateral.	Computation not performed	XV_CFI_ZONE_VIS_TIME_POINT_IN_CUADRILATERAL_ERR	40
ERR	Error sorting intersections.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SORT_INTERSECTIONS_ERR	41

ERR	Cannot (re)allocate memory for the segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SEGMENTS_MEMORY_ERR	42
ERR	Too many time segments (more than MAX_ORBITS).	Computation not performed	XV_CFI_ZONE_VIS_TIME_MAX_ORBITS_ER R	43
ERR	Cannot allocate memory for the coverage.	Computation not performed	XV_CFI_ZONE_VIS_TIME_COVERAGE_MEMORY_ERR	44
WAR N	Warning checking the visibility segments.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_CHECK_SEGMENTS_WARN	45
ERR	Error checking the visibility segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_CHECK_SEGMENTS_ERR	46
ERR	Error computing final segments for the POINT swath and POINT zone.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_TO_TIME_CALL_ERR	47
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_MODEL_E RR	48
WAR N	"stop_orbit" is after the last orbit in the orbit file.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_STOP_ORBIT_WA RN	49
ERR	Error computing the ANX longitude	Computation not performed	XV_CFI_ZONE_VIS_TIME_COMPUTE_ANX_ ERR	50
ERR	Error calling "orbit info"	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_INFO_ERR	51
ERR	Error computing Multi- Point swath visibilities	Computation not performed	XV_CFI_ZONE_VIS_TIME_MULTI_POINT_SWATH_INTERS_ERR	52
ERR	Error computing Point swath visibilities	Computation not performed	XV_CFI_ZONE_VIS_TIME_POINT_SWATH_INTERS_ERR	53
ERR	Error checking visibility segments	Computation not performed	XV_CFI_ZONE_VIS_TIME_ON_OFF_CHECKING_ERR	54
ERR	Error merging visibility segments	Computation not performed	XV_CFI_ZONE_VIS_TIME_MERGE_SWATH_SEGMENTS_VISIBILITIES_ERR	55
ERR	Error trying to allocate memory	Computation not performed	XV_CFI_ZONE_VIS_TIME_MEMORY_ALLOCATION_ERR	56
ERR	Error calling "swath_pos”	Computation not performed	XV_CFI_ZONE_VIS_TI	57

			ME_SWATH_POS_ER R	
ERR	Error calling "Polygon_inner_point"	Computation not performed	XV_CFI_ZONE_VIS_TI ME_POLYGON_INNE R_POINT_ERR	58
ERR	Error comparing orbits orbital changes	Computation not performed	XV_CFI_ZONE_VIS_TI ME_CHECK_ORBITAL_CH ANGE_ERR	59
ERR	Error converting zone point arrays to zone record	Computation not performed	XV_CFI_ZONE_VIS_TI ME_CONVERT_ZONE_ER R	64
ERR	No suitable zone found for orbit interval	Computation not performed	XV_CFI_ZONE_VIS_TI ME_ZONE_ORBIT_ERR	65
ERR	Error cloning zone	Computation not performed	XV_CFI_ZONE_VIS_TI ME_CLONE_ZONE_ERR	66
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_ZONE_VIS_TIME_ ORBIT_INTERVAL_STF_ER R	67
WARN	Input orbit interval is partially outside STF validity interval	Computation performed	XV_CFI_ZONE_VIS_TIME_ ORBIT_INTERVAL_STF_W ARN	68
ERR	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation not performed	XV_CFI_ZONE_VIS_TIME_ OSF_NON_LIN_STF_OLD_ WARN	69
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation performed	XV_CFI_ZONE_VIS_TIME_ SWATH_FLAG_LARGER_T HAN_LIN_APPROX_VAL_W ARN	70
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ GEO_SAT_ERR	71
WARN	This function is deprecated. Use xv_zonevistime_compute instead	Computation performed	XV_CFI_ZONE_VIS_TIME_ DEPRECATED_WARN	72
WARN	Visibility computations with precise propagator can be very slow	Computation performed	XV_CFI_ZONE_VIS_TIME_ PRECISE_PROPAG_WARN	73

Note that error codes and messages have been completely modified since the last issue due to a completely new implementation of the CFI function.

7.2 xv_zone_vis_time_no_file

7.2.1 Overview

Note: this function is deprecated. Use **xv_zonevistime_compute** instead.

The **xv_zone_vis_time_no_file** function computes all the orbital segments for which a given instrument swath intercepts a user-defined zone at the surface of the Earth ellipsoid.

The aim of this function is to provide another interface for the function **xv_zone_vis_time** in which the zone and the swath are not provided with files but with the data structures (see section 7.2.2).

Information about zones, swaths and intersection algorithms can be found in section 7.1.

7.2.2 Calling sequence

For C programs, the call to **xv_zone_vis_time_no_file** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
      xo_orbit_id orbit_id = {NULL};
      long     orbit_type,
               start_orbit, start_cycle,
               stop_orbit, stop_cycle,
               zone_num, projection,
               number_segments,
               *bgn_orbit, *bgn_second,
               *bgn_microsec, *bgn_cycle,
               *end_orbit, *end_second,
               *end_microsec, *end_cycle,
               *coverage, ierr[XV_NUM_ERR_ZONE_VIS_TIME],
               status;
      double   *zone_long, *zone_lat,
               zone_diam, min_duration;
      xd_stf_file  stf_data;
      xd_zone_rec  zone_data;

      status = xv_zone_vis_time_no_file(&orbit_id,
                                 &orbit_type,
                                 &start_orbit, &start_cycle,
                                 &stop_orbit, &stop_cycle,
                                 &stf_data,
                                 &zone_data,
                                 &projection, &zone_num,
```

```
                                   zone_long, zone_lat, &zone_diam,
                                   &min_duration,
                                   &number_segments,
                                   &bgn_orbit, &bgn_second,
                                   &bgn_microsec, &bgn_cycle,
                                   &end_orbit, &end_second,
                                   &end_microsec, &end_cycle,
                                   &coverage,   ierr);

        /* Or, using the run_id */
        long run_id;

        status = xv_zone_vis_time_no_file_run(&run_id,
                                   &orbit_type,
                                   &start_orbit, &start_cycle,
                                   &stop_orbit, &stop_cycle,
                                   &stf_data,
                                   &zone_data,
                                   &projection, &zone_num,
                                   zone_long, zone_lat, &zone_diam,
                                   &min_duration,
                                   &number_segments,
                                   &bgn_orbit, &bgn_second,
                                   &bgn_microsec, &bgn_cycle,
                                   &end_orbit, &end_second,
                                   &end_microsec, &end_cycle,
                                   &coverage,   ierr);
}
```

7.2.3 Input parameters

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

Table 10: Input parameters of xv_zone_vis_time_no_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long*	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters. Relative orbits only can be used when the orbit_id was initialized with orbital changes (with xo_orbit_init_def or with xo_orbit_init_file plus an OSF file). In other cases, only the value XV_ORBIT_ABS can be used.	-	Complete.
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit. First Orbit for the orbit initialization will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤repeat cycle
• start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using absolute orbits	cycle number	= 0 or ≥first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. For orbit_id initialized with orbital changes, when: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) or • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤repeat cycle

			change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle) If it is not initialized with orbital changes, stop orbit will be set to the last orbit in orbit_id initialization.		
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using absolute orbits	cycle number	= 0 or ≥ first cycle in osf
stf_data	xd_stf_file	-	Swath template data (structure described in [D_H_SUM]). The swath structure can be got by: • Reading a swath template file with the CFI function **xd_read_stf**. • Generating the swath data with the CFI function **xv_gen_swath_no_file**	-	-
zone_data	xd_zone_read	-	Zone data (structure described in [D_H_SUM]) that can be got by reading a zone from a zone database file with the CFI function **xd_read_zone**.	-	-
projection	long		projection used to define polygon sides as straight lines: = 0 Read projection from Zones DB = 1 Azimuthal gnomonic = 2 Rectangular lat/long	-	-
zone_num	long		Number of vertices of the zone provided in zone_long, zone_lat: = 0 no vertices provided, use zone_id / zone_db_file = 1 Point / Circular zone, = 2 Line zone > 2 Polygon zone		≥ 0
zone_long	double*	all	zone_long[i-1] Geocentric longitude of - circle centre, for circ. zone, i =1 - point, for point zone, i = 1		

			- line-end, for line zone, i = 1 or 2 - vertices, for polygon zone, i = 1... zone_num		
zone_lat	double*	all	zone_lat[i-1] Geodetic latitude of - circle centre, for circ. zone, i =1 - point, for point zone, i = 1 - line-end, for line zone, i = 1 or 2 - vertices, for polygon zone, i = 1... zone_num		
zone_diam	double		Zone diameter for circular zones, dummy for other zones If diameter equals 0.0 then zone is Point Zone	m	≥ 0.0
min_duration	double		Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0

It is also possible to use enumeration values rather than integer values for some of the input arguments, as shown in the table below:

Input	Description	Enumeration value	long
projection (defined in [D_H_SUM])	Read projection from the zones DB file	XD_READ_DB	0
	Azimuthal Gnomonic	XD_GNOMONIC	1
	Rectangular long/lat	XD_RECTANGULAR	2

7.2.4 Output parameters

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

Table 11: Output parameters of xv_zone_vis_time_no_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xv_zone_vis_time_no_file	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long		Number of visibility segments returned to the user.		≥ 0
bgn_orbit	long*	all	Orbit number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		> 0
bgn_second	long*	all	Seconds since ascending node, begin of visibility segment i bgn_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period
bgn_microsec	long*	all	Micro seconds within second begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments	µs	≥0 999999
bgn_cycle	long*	all	Cycle number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
end_orbit	long*	all	Orbit number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		> 0
end_second	long*	all	Seconds since ascending node, end of visibility segment i end_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period

end_microsec	long*	all	Micro seconds within second end of visibility segment i end_microsec[i-1], i = 1, number_segments	µs	≥ 0 999999
end_cycle	long*	all	Cycle number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
coverage	long*	all	Zone coverage flag for segment = 0 Zone completely covered by swath = 1 Zone not completely covered by swath, extending over the left edge of the swath. = 2 Zone not completely covered by swath, extending over the right edge of the swath. = 3 Zone not completely covered by swath, extending over both edges of the swath coverage[i], i = 0, (number_segments-1)		
ierr[XV_NUM_ERR_ZONE_VIS_TIME]	long		Error status flags		

It is also possible to use enumeration values rather than integer values for some of the output arguments, as shown in the table below:

Input	Description	Enumeration value	long
coverage	Zone completely covered by swath	XV_COMPLETE	0
	Left extreme transitions found	XV_LEFT	1
	Right extreme transitions found	XV_RIGHT	2
	Both extreme transitions found	XV_BOTH	3

<u>Memory Management:</u> Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_zone_vis_time_no_file** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.2.5 Warnings and errors

The error and warning messages and codes for **xv_zone_vis_time_no_file** are the same than for **xv_zone_vis_time** (see Table 9) .

The error messages/codes can be returned by the CFI function **xv_get_msg/xv_get_code** after translating the returned status vector into the equivalent list of error messages/codes. The function identifier to be used in that functions is XV_ZONE_VIS_TIME_ID (from Table 2).

7.3 xv_zonevistime_compute

7.3.1 Overview

The **xv_zonevistime_compute** function computes all the orbital segments for which a given instrument swath intercepts a one or more user-defined zones at the surface of the Earth ellipsoid.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as seconds (and microseconds) elapsed since the ascending node crossing.

A user-defined zone can be:

- a polygon specified by a set of latitude and longitude points
- a circle specified by the centre latitude, longitude, and the diameter

Note that particular cases of the above can be used to define the zone as:

- a point
- a line

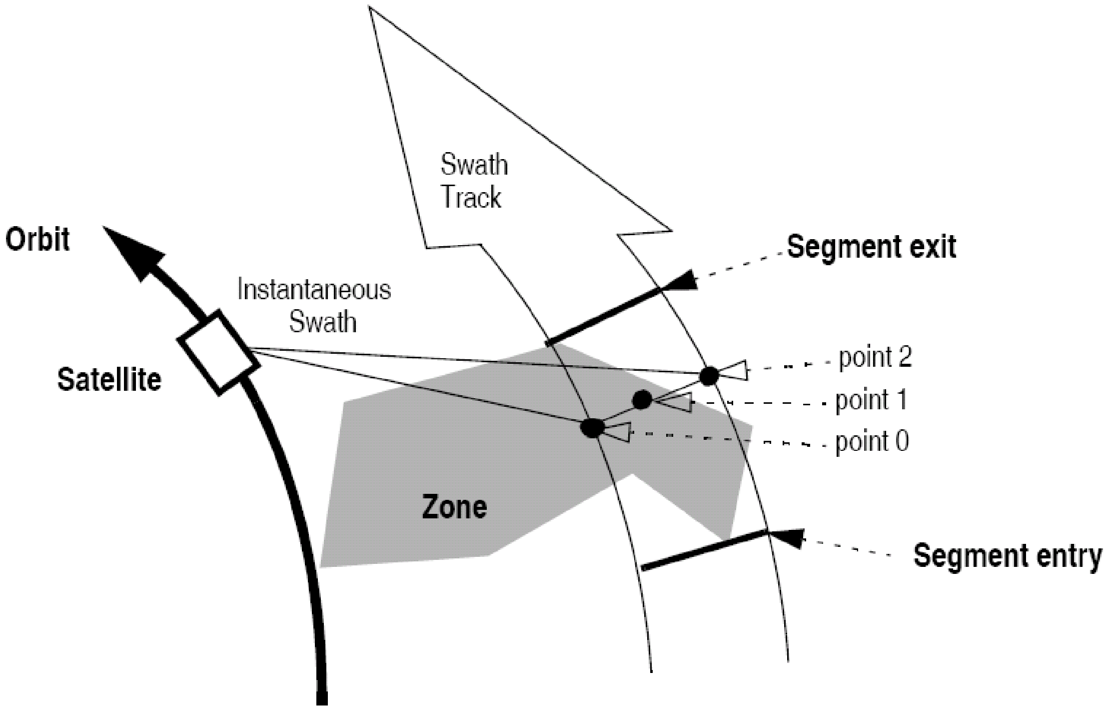


Figure 10: Segment Definition xv_zonevistime_compute

If more than one zone is used as input, the visibilities are internally computed for each zone, and the segments are merged and ordered by start time. In the output visibility list, the zones where the segment has visibility are provided, and also the coverage of the segment for each zone (see Figure 11).

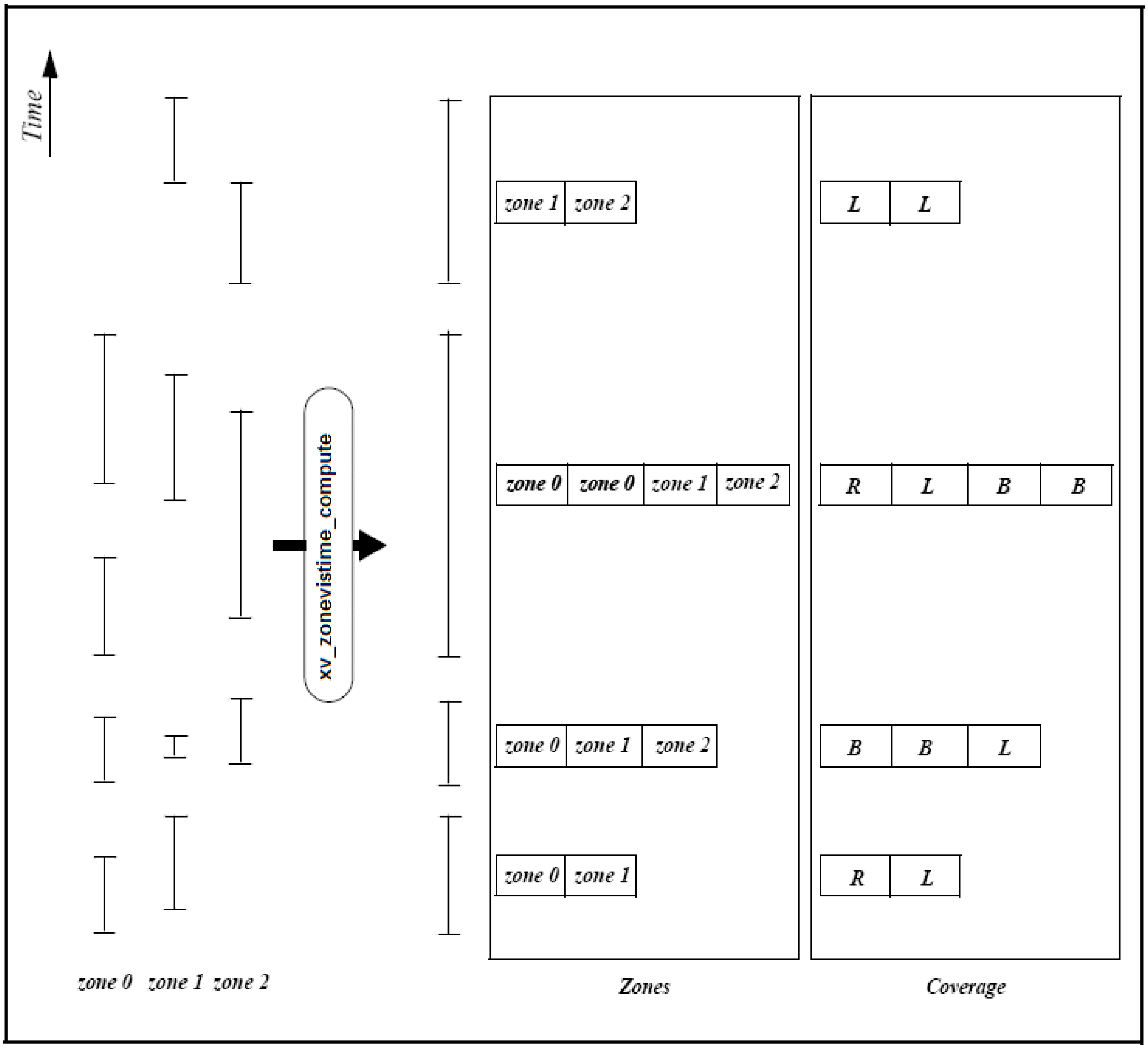


Figure 11: xv_zonevistime_compute function (more than one zone)

xv_zonevistime_compute requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files, also with precise propagator initialization if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**).
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
  - TLE files

Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.

- the swath_id (xv_swath_id, initialized using xv_swath_id_init -section 7.32-) providing the Instrument Swath information, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit. If the swath_id is initialized with a Swath definition file or Swath definition data, **xv_zonevistime_compute** generates the swath points for a number of orbits given by the user.
- The information of the zone or zones (xv_zone_info_list)**.**

The time intervals (xv_time_interval) used by **xv_zonevistime_compute** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times. Moreover, the segments will be ordered chronologically.

NOTE: If **xv_zonevistime_compute** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behavior depends on the type of the data used to initialize the swath_id (via xv_swath_id_init, section 7.32):

•If a **swath template file** is used, **xv_zonevistime_compute** automatically will ignore the orbits that do not correspond with the template file (i.e. no visibility segments will be generated for those orbits), since swath template file is generated from a reference orbit with a particular geometry, so it is not valid for a different geometry.

•If a **swath definition file** is introduced, **xv_zonevistime_compute** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath template files valid for every orbital change are generated to perform the calculations.

NOTE 2:If a swath template file with the variable header tags *Start_Validity_Range* and *Stop_Validity_Range* is used as input, only the segments belonging to that orbit range will be returned.

NOTE 3: If a swath definition file is introduced, it can be also introduced every how many orbits the swath template file must be recomputed (the number of orbit for regeneration is introduced in swath_id initialization). If the orbit_id has been initialized with an OSF file with MLST non linear terms and the number of orbits for regeneration is greater than the linear approximation validity, the recomputation of swath template file will be done every linear approximation validity orbits.

7.3.2 Swath Definition

The swath file that can be used to initialize the swath id is generated using the xv_gen_swath function, within the EO_VISIBILITY library. There are 3 different types of swaths:

- earth-observing instruments ('nadir curve', 'nadir point' or "area swaths")
- limb-sounding instruments ('limb', narrow or wide)
- limb-sounding instruments observing inertial objects ('inertial')

The following sub-sections provide some details on the various swath definitions.

7.3.2.1 Earth-observing Instruments Swath Definition

The term swath must be clearly defined to understand the explanations in this document:

- instantaneous swath: the part of the earth surface observed by an instrument at a given time
- swath track: represents the track made on the earth surface by the instantaneous swath over a period of time

For instruments observing the surface of the earth, the instantaneous swath is constituted by the point/curve/area on the ground observed by the instrument at a given time. It is calculated taking the earth ellipsoid as a reference for the earth surface. The wider the field-of-view of the instrument, the wider the swath on the ground.

When the satellite moves over a period of time, this point/curve/area defines a band on the earth surface. This constitutes the swath track.

See next figure for an illustration of these definitions.

Note that the terms curve or point are an idealized view of the instrument FOV, which usually have a thickness.

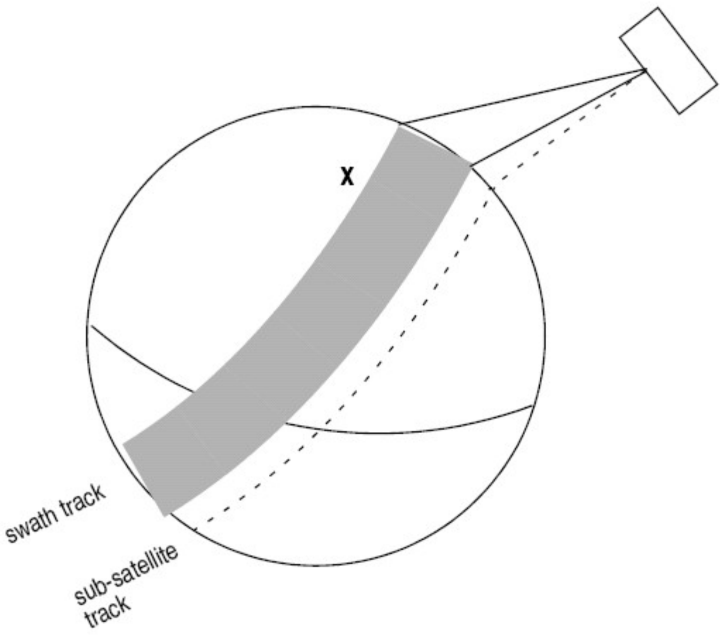


Figure 12: Earth-observing instrument: swath definition

7.3.2.2 Limb-sounding Instruments Swath Definition

For limb sounding instruments, the concept can be generalized to define a "thick swath". This is obtained by defining a minimum and a maximum altitude, and considering the tangent points to these altitudes as the edges of the swath. Two cases have to be considered:

- deterministic (narrow) azimuth field of view (e.g. MIPAS sideward-looking): the swath projection on the earth surface is similar to a regular sideward-looking swath, with the lower altitude defining the further swath edge and the higher altitude defining the closer swath edge. See Figure 13.
- non-deterministic (potentially wide) azimuth field of view (e.g. MIPAS rearward-looking): due to the potentially wide azimuth field of view, each altitude defines a swath projection on the earth surface. Depending on the altitude, these swaths are of different width across-track, and also at different distance from the satellite. See Figure 14.

For these, 2 Instrument Swath Files are provided:

- one at the highest altitude
- one at the lowest altitude

The user must handle both swath himself to determine his required visibility time segments

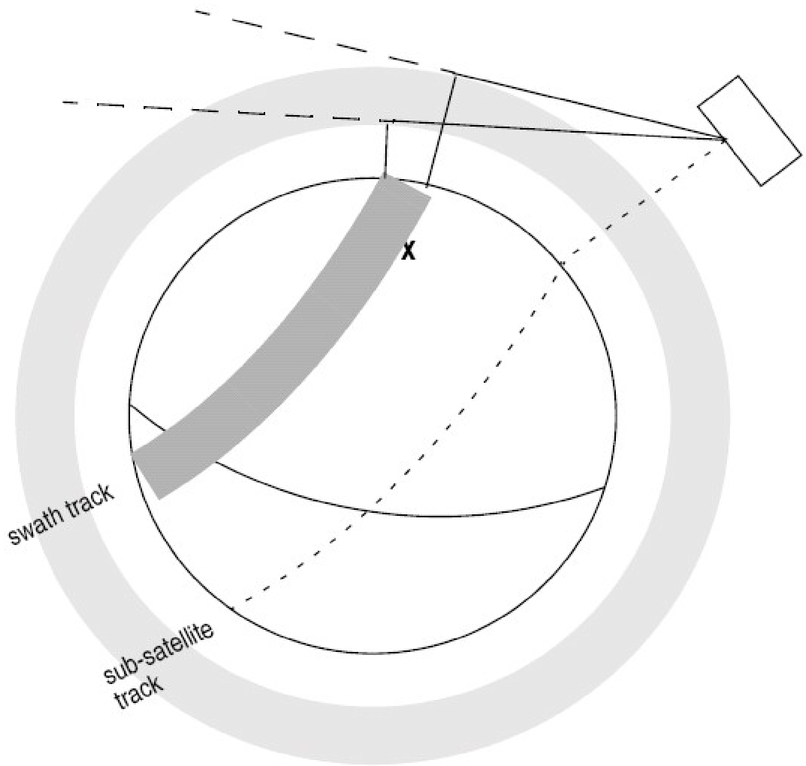


Figure 13: Limb-sounding instrument: swath definition (1)

.

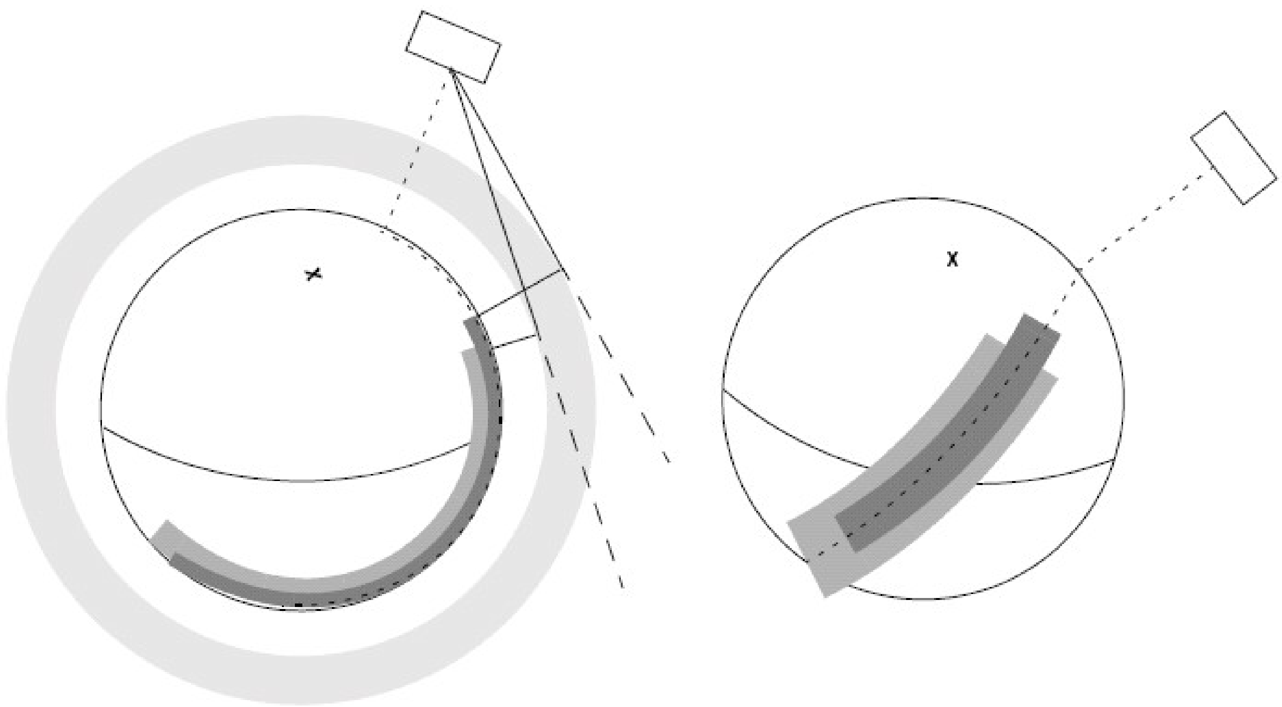


Figure 14: Limb-sounding instrument: swath definition (2)

7.3.2.3 Limb-sounding Instruments Inertial Swath Definition

This type corresponds to the observation of inertial targets (e.g. Gomos occultation mode and Mipas Line of Sight mode in Envisat). For the CFI function **xv_star_vis_time** the FOV direction in inertial coordinates must be available. Therefore for these instrument modes the direction in inertial space, for a given tangent altitude, is given in the swath template file.

7.3.2.4 Swath Definition for Envisat

Next table lists all instrument modes and the relevance of the swaths for Envisat. It shows also:

- the prefix to be used when generating the swath template file name
- the different types of algorithms to be used by xv_gen_swath (this is transparent to the user)

Table 12: Envisat Swaths

instrument	Mode	File Prefix = swath	Swath geometry (Table 90)	Swath Type	Remarks
RA		RA_2__	POINTING (1 point)	Nadir point	Modeled as sub-satellite track
MERIS	Averaging / Direct & Averaging	MERIS_	POINTING (3 points)	Nadir line	
ASAR	Image Modes (IS1... IS7)	SARxIM (x=1...7)	ASAR	Nadir line	
	Alt. Polarization (IS1... IS7)				
	Wide Swath	SARWIM			
	Global Monitoring				
	Wave (IS1... IS7)	SARxWV (x=1...7)			Modeled as a continuous swath any where within the image swath
GOMOS	Occultation	GOMOIL GOMOIH	INERTIAL	Inertial direction	IFOV much smaller than swath. IFOV Very dependent on star avail ability. 2 swaths defined: - 1 for high altitude (GOMOIH) - 1 for low altitude (GOMOIL)
	Occultation	GOMO_H GOMO_L	LIMB	Limb wide	Same mode as above, now swath defined as Earth-fixed location. IFOV much smaller than swath. IFOV Very dependent on star avail ability.

					2 swaths defined: - 1 for high altitude (GOMO_H) - 1 for low altitude (GOMO_L)
SCIAMACHY	Nadir / Nadir of Nadir & Limb	SCIAN_	POINTING (3 points)	Nadir line	Continuous Nadir swath modeled
	Limb / Limb of Nadir & Limb	SCIALH SCIALL		Limb wide	2 swaths defined: - 1 for high altitude (SCIALH) - 1 for low altitude (SCIALL)
AATSR		ATSR_N ATSR_F	POINTING (3 points)	Nadir line	2 swaths defined: - 1 for nadir swath - 1 for forward swath
MWR		MWR___	POINTING (1 points)	Nadir point	Modeled as sub-satellite track
MIPAS	Nominal	MIPN_H MIPN_L	LIMB	Limb narrow	2 swaths defined: - 1 for high altitude (MIPN_H) - 1 for low altitude (MIPN_L)
	Special Event Mode (across)	MIP_X_	LIMB	Limb narrow	Modeled as an across track swath, in the middle of the MIPAS SEM acquisition scan.
	Special Event Mode (rearward)	MIP_RH MIP_RL	LIMB	Limb wide	IFOV much smaller than swath. 2 swaths defined: - 1 for high altitude (MIP_RH) - 1 for low altitude (MIP_RL)
	Rearward Sideward	MIPIRH MIPIRL MIPIXH MIPIXL	INERTIAL	Inertial direction	2 swaths defined for rearward mode: - 1 for high altitude (MIPIRH) - 1 for low altitude (MIPIRL) 3 swaths defined for sideward mode: - 1 for high altitude (MIPIXH) - 1 for back mode (MIPIXB) - 1 for forward mode (MIPIXF)

7.3.3 Zone Borders and Projection

When defining a polygon zone, the user is assumed to wish polygon sides as straight lines. But on the earth surface, a straight line is, at best, a confusing concept.

The only way to define unambiguously straight lines is to work in a 2-dimensional projection of the earth surface. There are many possible projections, each having advantages and drawbacks.

xv_zonevistime_compute can handle zone borders in 2 different projections:

- rectangular projection, using longitude and latitude as the X and Y axis; this is appropriate to express zones where (some of) the edges follow constant latitude lines, and provide a reasonable approximation for straight lines at low-medium latitudes
- azimuthal gnomonic projection, where great circles are always projected as straight lines; this is better for high latitudes, where the rectangular projection suffers from too much distortion and the singularity at the poles.

xv_zonevistime_compute allows the user to specify which projection he wants to work in, i.e. in which projection the polygon sides will be represented by **xv_zonevistime_compute** as straight lines. The user is assumed to be aware of how the polygon sides behave on the Earth surface.

7.3.4 Zone Definition

The user-defined zone can be either (see Table 13);

- a point
- a line
- a polygon
- a circle

A zone is defined by the area of the earth surface enclosed by the zone borders:

- in the case of a circular zone, the area inside the circle
- in the case of a polygonal zone, the area which is always to the right of any polygon side; if the polygon is defined as a sequence of N points, each polygon side is considered as a line from point i to point i+1; this unambiguously defines the right side of the polygon sides.

Table 13: Zone definition (for xd_zone_rec)

Zone definition	num_points	zone_point (long, lat)	zone_diam	Description
Circular Zone	1	[0]: centre point	yes zone_diam > 0.0	The zone is represented as a circle, around the centre point
Point Zone	1	[0]: Point	yes zone_diam = 0.0	The zone is defined by the point. Resulting segments will have a zero duration. The zone will always be completely covered by the swath.
Line Zone	2	[0], [1]: Line	no	The zone is defined by the line from point [0] to point [1].
Polygon Zone	>2	[i]	no	The zone is defined by the area right of the line from point [i] to point [i+1].

For the gnomonic projection, a side of a zone is always smaller than a half great circle, because two polygon points are considered to be joined by the shortest line.

For the rectangular projection, two consecutive points of the zone are also joined by the shortest line; so the difference in longitude must be less than 180 degrees.

The polygon zone can be closed (i.e. the first and last points are the same) or not. If the zone is not closed, **xv_zonevistime_compute** closes it by joining the last point with the first one in its internal computations.

See Figure 15 for examples of zone definitions.

xv_zonevistime_compute will issue an error on the zone definition if the polygon has intersecting sides ("butterfly" zone).

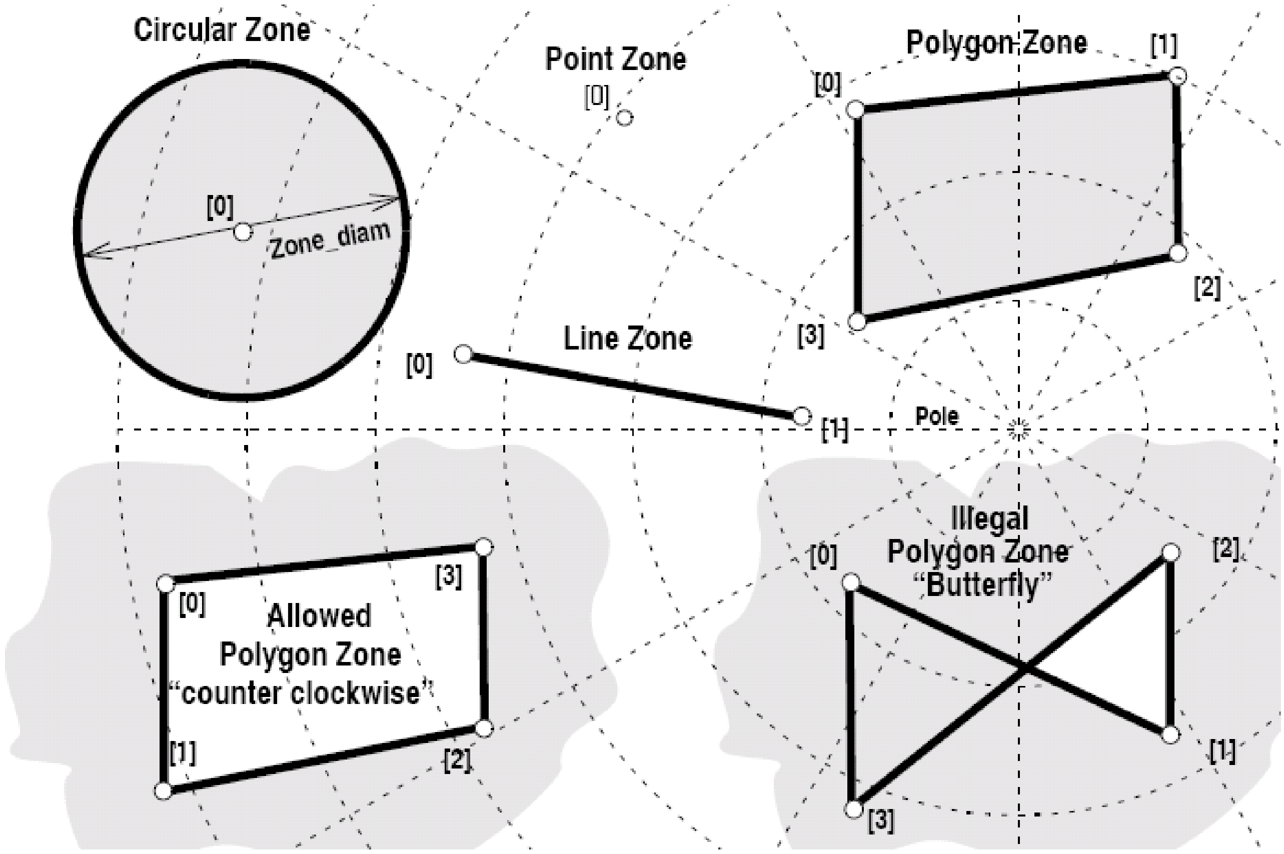


Figure 15: Zone examples

7.3.5 Intersection Definition

The **xv_zonevistime_compute** intersection times between the instrument swath and the user-defined zone are defined as the first and last occurrence, in chronological order with respect to the satellite direction, of the geometrical super-position of any point belonging to the instrument swath with any single point belonging to the zone (including the zone border).

The entry and exit times for each intersection are given as elapsed seconds (and microseconds) since the ascending node crossing.

Next figure shows some typical intersections.

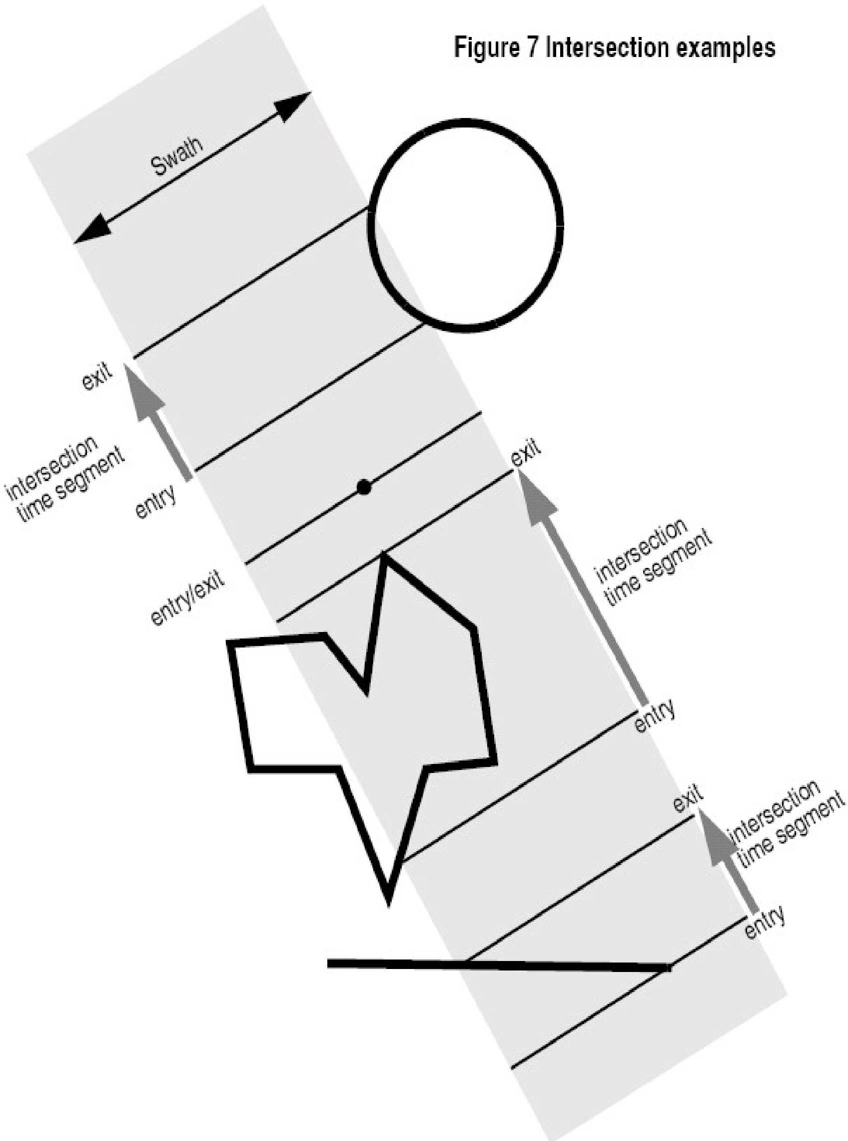


Figure 16: Intersection examples

7.3.6 Intersection Algorithm

The intersection of a swath and a user-defined zone is to be performed on the Earth projected to a map plane in one of the following projections:

- Rectangular projection
- Gnomonic projection

Although the projections are quite different, the intersection rules are identical. The algorithm can however be different, in order to take advantage of a particular feature of a projection.

The purpose of the CFI function ZONEVISTIME_COMPUTE is to obtain quickly, accurate intersection segments with a low precision (1 second).

The algorithms assume that the polygon zones are closed and expects a wrap around between the first and the last point. Thus ZONEVISTIME_COMPUTE must first close the polygon if necessary.

For ZONEVISTIME the following swath types are defined:

- point swath: instantaneous swath is a point.
- segment swath: instantaneous swath is a segment.
- multi-segment swath: it can be open or closed.
- inertial swath: not used by ZONEVISTIME

The main concept in the algorithm is the transition, defined as the change in coverage of (part of) the swath and the zone (e.g. edge of the swath crosses one polygon side).

7.3.6.1 Intersection with a point swath

The vertices of the polygon defining the area are connected by straight lines in the chosen projection, along track swath points are also connected by straight lines in the same projection.

Transitions are located by linear intersection of the zone sides and the swath along track lines. A transition is only valid if the intersection occurs inside both line segments. The polygon side from <i> to <j> is defined in a clockwise manner inclusive point <i> but exclusive point <j>. The swath line from time <k> to <l> is defined inclusive the template point at <k> but exclusive the template point at <l>.

The fraction of the swath along track line determines the precise timing since time <k> of the intersection. Also the determination if the transition is a on- or off-transition is quite trivial. First a vector is defined, perpendicular to the along track swath line, such that the vector points left. Then, the dot product of the polygon side and this vector is calculated. If the dot product is positive, the transition is on, i.e. the swath enters the zone. If the result is negative, then the swath leaves the zone. If the result equals zero then the transition can be ignored (polygon side and swath overlay, a proper transition will be found with another pair of polygon side - swath line.).

7.3.6.2 Intersection with a segment swath

The left and right side of the swath, are located using the same algorithm as for the point swath. Even left and right time segments can be made based on the left and right hand transitions.

The polygon vertices (and not the sides) are intersected with the along track moving line swath, in order to catch zones smaller than the swath, etc. Swaths for intermediate times between two consecutive times in Swath Template File are considered straight segments, joining an intermediate point of the Left swath line from time <k> to time <l>, with an intermediate point in Right swath line.

7.3.6.3 Intersection with a multi-segment swath

The algorithm used for segment swath is repeated for every segment of the swath, and the visibility segments obtained in each case are merged with the ones of the other swath segments.

For a closed swath further calculations are done: it is checked if the zone is completely inside the swath area in the interval between contiguous visibility segments, or between the beginning of the first orbit and the first visibility segment, or between the last visibility segment and the end of the last orbit computed. If it is inside, segments must be merged because the zone was visible in the interval.

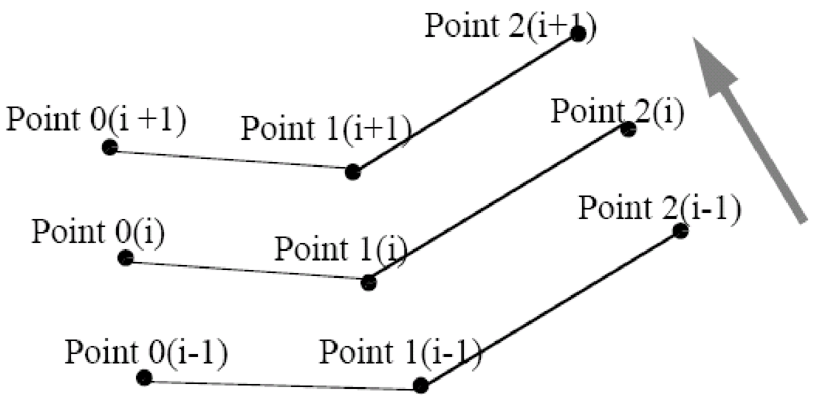


Figure 17: Swath points

7.3.7 Usage Hints

7.3.7.1 Limb-sounding Instruments Intersection

In the case of limb-sounding instrument with a potentially wide azimuth field of view, 2 swaths have to be considered (1 for minimum altitude, 1 for maximum altitude). Furthermore, these 2 swaths are offset in time (i.e. their projection on the earth intersect with a given point at different times). To cope with this, the user must do the following:

- call **xv_zonevistime_compute** twice (once for each extreme altitude swath)
- merge/filter the 2 sets of time segments, depending on what he wants to achieve

7.3.7.2 Zone Coverage

xv_zonevistime_compute computes purely geometrical intersections. The resulting zone visibility segments might need some additional filtering by the user. In particular, instrument constraints (e.g. only working outside of sun eclipse) have to be considered by the user.

Furthermore, to help users to deal with zones wider than the swath (i.e. requiring several orbits to cover the whole zone), **xv_zonevistime_compute** produces for each zone visibility segment an indication of the coverage type (see Figure 18) and the name of the zone (or a lists in case several zones are seen by that segment9;

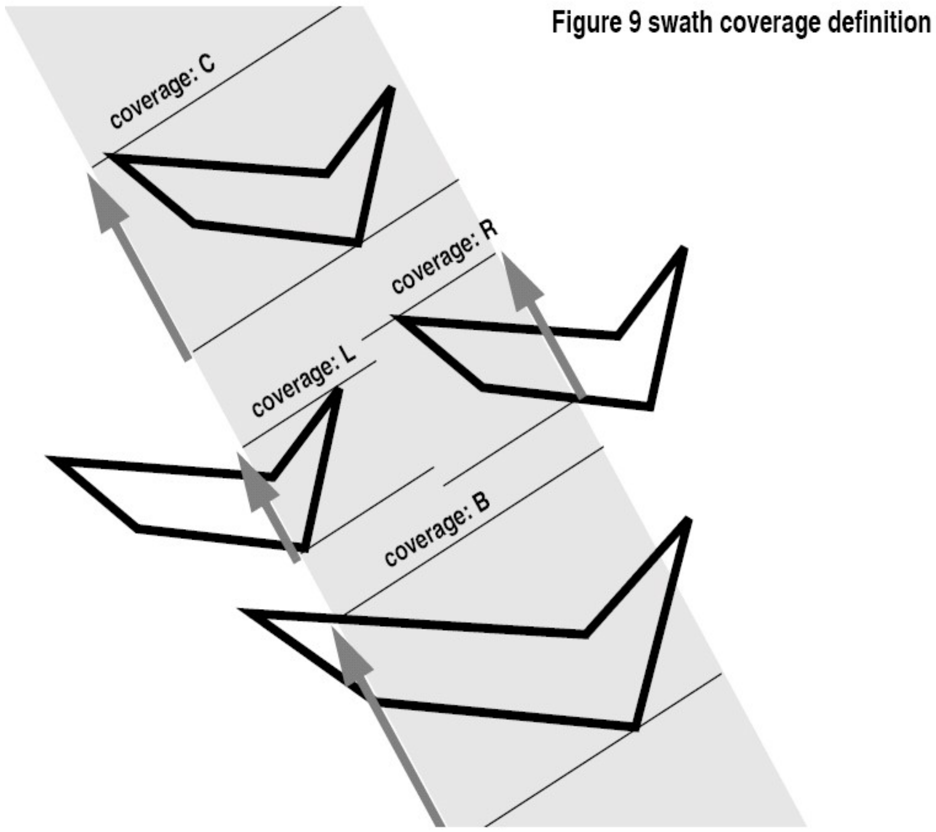


Figure 18: Swath coverage definition

- coverage = C: zone completely covered by the swath
- coverage = R: zone partially covered by the swath, extending over the right edge of the swath
- coverage = L: zone partially covered by the swath, extending over the left edge of the swath
- coverage = B: zone partially covered by the swath, extending over both edges of the swath

7.3.7.3 Combined use of *xv_swathpos_compute and the coverage flag*

The EO_VISIBILITY function xv_swathpos_compute can be used to refine the work performed with **xv_zonevistime_compute**.

7.3.7.4 Use of input xp_attitude_def struct

The definition of the input structure xp_attitude_def can be consulted in section 6.3 of [POINT_SUM].

The "type" field defines how this struct is used, and it can take the following values:

- XP_NONE_ATTITUDE: no attitude defined in struct. In this case, when the Swath Template File must be computed internally, the attitudes defined in Swath Definition File are used.
- XP_SAT_NOMINAL_ATT, XP_SAT_ATT, XP_INSTR_ATT: the attitudes defined in the structure are used in internal Swath Template File generation. The "type" field in this case indicates the target frame for the computation.

7.3.7.5 Use of input xv_zone_info_list struct

The zone or zones to be used in algorithm are passed to xv_zonevistime_compute function with the struct *xv_zone_info_list* (see section 6.3 for description). It contains the following fields:

- *calc_flag*: it can take the enumeration values XV_COMPUTE or XV_DO_NOT_COMPUTE. This flag indicates if the extra information regarding the zones (coverage) must be computed or not.
- *num_rec*: Indicates the number of input zones where the visibility is going to be computed.
- *zone_info*: description of the zones to be computed. Every position of the array is a *xv_zone_info* struct . The value of *type* field indicates the type of zone data:
  - If *type* is equal to enum value XV_USE_ZONE_DB_FILE, then the zone *zone_id* is read from *zone_db_filename* zone file and *projection* projection (possible values are the following enums: XD_READ_DB, XD_GNOMONIC or XD_RECTANGULAR) is used.
  - If *type* is equal to enum value XV_USE_ZONE_DATA, then the information contained in *zone_data* field is used (see [D_H_SUM] for description of *xd_zone_rec*).
  - *min_duration*: indicates the minimum duration for the segments (seconds).

7.3.8 Calling sequence

For C programs, the call to **xv_zonevistime_compute** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xo_orbit_id orbit_id = {NULL};
       xp_attitude_def attitude_def;
       xv_swath_id swath_id = {NULL};
       xv_zone_info_list zone_info_list;
       xv_time_interval search_interval;
       xv_zonevisibility_interval_list visibility_interval_list;
       long ierr[XV_NUM_ERR_ZONEVISTIME_COMPUTE];

       status = xv_zonevistime_compute(&orbit_id,
                             &attitude_def,
                             &swath_id, &zone_info_list,
                             &search_interval,
                             &visibility_interval_list, ierr);

}
```

7.3.9 Input parameters

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

Table 14: Input parameters of xv_zonevistime_compute function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
attitude_def	xp_attitude_def*	-	Structure with the definition of the attitudes.	-	-
swath_id	xv_swath_id*	-	Swath id.	-	-
zone_info_list	xv_zone_info_list*	-	List of zones where the visibility is going to be computed.	-	-
search_interval	xv_time_interval*	-	Interval where the computations are performed	-	-

7.3.10 Output parameters

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

Table 15: Output parameters of xv_zonevistime_compute function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xv_zonevistime_compute	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
visibility_interval_list	xv_zonevisibility_interval_list*	-	List of visibility segments and additional information	-	-
ierr[XV_NUM_ERR_ZONEVISTIME_COMPUTE]	long		Error status flags		

It is also possible to use enumeration values rather than integer values for some of the output arguments, as shown in the table below:

Input	Description	Enumeration value	long
coverage	Zone completely covered by swath	XV_COMPLETE	0
	Left extreme transitions found	XV_LEFT	1
	Right extreme transitions found	XV_RIGHT	2
	Both extreme transitions found	XV_BOTH	3

Memory Management: Note that the output visibility segment list (xv_zonevisibility_interval_list->visibility_interval) is a pointer to the list of segments computed inside xv_zonevistime_compute. The memory for these dynamic array is allocated within the **xv_zonevistime_compute** function. So the user will only have to declare the variable xv_zonevisibility_interval_list. However, once the function has returned without error, the user will have the responsibility of freeing the memory for the pointers inside that variable and the rest of structs inside xv_zone_coverage_info_list struct.

7.3.11 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_zonevistime_compute** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_zonevistime_compute** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code.**

Table 16: Error messages and codes for xv_zonevistime_compute

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Input parameter \"Number of ZONE points\" is wrong.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_NEGATIVE_NUM_ZONE_ERR	0
ERR	Input parameter \"Orbit Id\" is wrong.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBIT_STATUS_ERR	1
ERR	Input parameter \"orbit_type\" is out of range.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBIT_TYPE_ERR	2
ERR	Input parameter \"Minimum duration\" cannot be negative.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_NEGATIVE_MIN_DURATION_ERR	3
ERR	Input parameter \"Projection\" out of range.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_PROJECTION_OUT_OF_RANGE_ERR	4
ERR	Swath id not initialized	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_SWATH_STATUS_ERR	5
ERR	Swath file is not compatible with the orbit file	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_WRONG_SWATH_ERR	6
ERR	Error reading Swath File: %s	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_READ_SWATH_FILE_ERR	7
ERR	Input parameter \"start_orbit\" cannot be negative.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_NEGATIVE_START_ORBIT_ERR	8
WARN	\"start_orbit\" is before the first orbit in \"orbit_event_file\".	Computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_EARLIER_START_ORBIT_WARN	9

ERR	Input parameter \"start_orbit\" cannot be greater than \"stop_orbit\".	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_WRONG_ORBIT_RANGE_ERR	10
ERR	Error calling \"xv_orbitinfo\".	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBITINFO_CALL_ERR	11
ERR	Input parameter \"zone_id\" is an empty string.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ZONE_ID_EMPTY_ERR	12
ERR	Error reading the ZONE Database file: %s	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_READ_ZONE_DB_FILE_ERR	13
ERR	Cannot (re)allocate memory for the segments.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_SEGMENTS_MEMORY_ERR	14
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBIT_MODEL_ERR	15
WARN	\"stop_orbit\" is after the last orbit in the orbit file.	Computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_STOP_ORBIT_WARN	16
ERR	Error calling \"orbit info\"	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBIT_INFO_ERR	17
ERR	Error cloning zone	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_CLONE_ZONE_ERR	18
ERR	Input orbit interval is completely outside STF validity interval	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBIT_INTERVAL_STF_ERR	19
WARN	Input orbit interval is partially outside STF validity interval	Computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBIT_INTERVAL_STF_WARN	20
WARN	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_OSF_NON_LIN_STF_OLD_WARN	21
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_SWATH_FLAG_LARGER_THAN_LIN_APPROX_VAL_WARN	22
ERR	Geostationary satellite not allowed for this function.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_GEO_SAT_ERR	23
ERR	Error computing overlap for multizones	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_OVERLAP_ERR	24

ERR	Error computing zonevistime loop	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ZONEVISTIME_LOOP_ERR	25
ERR	Two ZONE segments intersect.	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_TWO_SEGMENTS_INTERSECT_ERR	26
WARN	There is at least one orbital change within the requested orbit range.	Computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_ORBITAL_CHANGE_WARN	27
ERR	\"cycle_length\" read from the input \"Swath Template File\" is not equal to that of any orbits within the orbit range	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_INCONSISTENT_SWATH_ERR	28
ERR	Input time type is not correct	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_TIME_TYPE_ERR	32
ERR	Error checking output segments	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_CHECK_SEGMENTS_ERR	33
ERR	Error transforming time to orbit	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_TIME_TO_ORBIT_ERR	34
ERR	Error computing UTC time	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_GET_UTC_TIME_ERR	35
ERR	Input file is not a swath file	No computations performed.	XV_CFI_ZONEVISTIME_COMPUTE_DETECT_SWATH_TYPE_ERR	36
WARN	Visibility computations with precise propagator can be very slow	Computation performed	XV_CFI_ZONEVISTIME_COMPUTE_PRECISE_PROPAG_WARN	37

7.4 xv_station_vis_time

7.4.1 Overview

Note: this function is deprecated. Use **xv_stationvistime_compute** instead.

The **xv_station_vis_time** function computes ground station visibility segments, the orbital segments for which the satellite is visible from a ground station located at the surface of the Earth.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as seconds elapsed since the ascending node crossing.

In addition, **xv_station_vis_time** calculates for every visibility segment the time of zero-doppler (i.e. the time at which the range-rate to the station is zero).

xv_station_vis_time requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data and files, also for precise propagation if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**).
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
  - TLE files

    Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.
- the Instrument Swath File, describing the area seen by the relevant instrument all along the current orbit. The Swath data can be provided by:
  - A swath template file produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).
  - A swath definition file, describing the swath geometry. In this case the **xv_station_vis_time** generates the swath points for a number of orbits given by the user.
- The Station Database File, describing the location and the physical mask of each ground station, and the mask parameters for a list of spacecrafts from each station (considered only when mask 'from file' option is selected).

The time intervals used by **xv_station_vis_time** are expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter “Orbit Range”: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle number should be used, otherwise, this the cycle number will be a dummy parameter.

- output parameter "Station Visibility Segments": time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

Users who need to use processing times must make use of the conversion routines provided in EO_ORBIT (**xo_time_to_orbit** and **xo_orbit_to_time** functions).

NOTE: If **xv_station_vis_time** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behaviour depends on the swath file introduced as input:

•If a **swath template file** is used, **xv_station_vis_time** automatically will ignore the orbits that do not correspond with the template file (i.e. no visibility segments will be generated for those orbits), since swath template file is generated from a reference orbit with a particular geometry, so it is not valid for a different geometry.

•If a **swath definition file** is introduced, **xv_station_vis_time** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath template files valid for every orbital change are generated to perform the calculations.

NOTE 2:If a swath template file with the variable header tags *Start_Validity_Range* and *Stop_Validity_Range* is used as input, only the segments belonging to that orbit range will be returned.

NOTE 3: If a swath definition file is introduced, it can be also introduced every how many orbits the swath template file must be recomputed (swath_flag parameter, see section 111). If the orbit_id has been initialized with an OSF file with MLST non linear terms and the parameter swath_flag is greater than the linear approximation validity, the recomputation of swath template file will be done every linear approximation validity orbits.

7.4.2 Calling interface

For C programs, the call to **xv_station_vis_time** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xo_orbit_id orbit_id = {NULL};
       long     swath_flag, orbit_type,
                start_orbit, start_cycle,
                stop_orbit, stop_cycle,
                mask, number_segments,
                *bgn_orbit, *bgn_second,
                *bgn_microsec, *bgn_cycle,
                *end_orbit, *end_second,
                *end_microsec, *end_cycle,
                *zdop_orbit, *zdop_second,
                *zdop_microsec, *zdop_cycle,
                ierr[XV_NUM_ERR_STATION_VIS_TIME],
                status;
       double   aos_elevation, los_elevation, min_duration;
       char     *swath_file;
       char     *sta_id, *sta_db_file;


       status = xv_station_vis_time(
                         &orbit_id, &orbit_type,
                         &start_orbit, &start_cycle,
                         &stop_orbit, &stop_cycle,
                         &swath_flag, &swath_file, sta_id, sta_db_file,
                         &mask, &aos_elevation, &los_elevation,
                         &min_duration,
                         &number_segments,
                         &bgn_orbit, &bgn_second,
                         &bgn_microsec, &bgn_cycle,
                         &end_orbit, &end_second,
                         &end_microsec, &end_cycle,
                         &zdop_orbit, &zdop_second,
                         &zdop_microsec, &zdop_cycle,
                         ierr);
```

```
        /* Or, using the run_id */
        long run_id;

        status = xv_station_vis_time_run(
                          &run_id, &orbit_type,
                          &start_orbit, &start_cycle,
                          &stop_orbit, &stop_cycle,
                          &swath_flag, &swath_file, sta_id, sta_db_file,
                          &mask, &aos_elevation, &los_elevation,
                          &min_duration,
                          &number_segments,
                          &bgn_orbit, &bgn_second,
                          &bgn_microsec, &bgn_cycle,
                          &end_orbit, &end_second,
                          &end_microsec, &end_cycle,
                          &zdop_orbit, &zdop_second,
                          &zdop_microsec, &zdop_cycle,
                          ierr);
}
```

7.4.3 Input parameters

Table 17: Input parameters of xv_station_vis_time

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤repeat cyc
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle) If it is not initialized with orbital changes, stop orbit will be set to the last orbit in orbit_id initialization.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤repeat cycle
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥first cycle in osf

swath_flag	long*	-	Define the use of the swath file: • 0 = (XV_STF) if the swath file is a swath template file. • > 0 if the swath files is a swath definition file. In this case the swath points are generated for every “swath_flag” orbits	-	XV_STF = 0 XV_SDF = 1 > 0
swath_file	char *	-	File name of the swath-file for the appropriate instrument mode		
sta_id	char*		identification name of the station		
station_db_file	char *		File name of the station database file This file is read each time the function is called		
mask	long		mask used to define visibility = XV_COMBINE combine AOS/LOS elevations and physical mask (nominal mode) = XV_AOS_LOS consider only AOS/LOS elevations = XV_PHYSICAL consider only physical mask = XV_FROM_FILE consider mask given in the Station Database File		all
aos_elevation	double		Minimum elevation to consider at AOS (i.e. before considering start of visibility). Not used if mask=XV_FROM_FILE	deg	≥ 0.0
los_elevation	double		Maximum elevation to consider at LOS (i.e. before considering end of visibility). Not used if mask=XV_FROM_FILE	deg	≥ 0.0 aos_elevation
min_duration	double		Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0.0

It is also possible to use enumeration values rather than integer values for some of the input arguments, as shown in the table below:

Input	Description	Enumeration value	long
mask	Combine AOS/LOS and physical mask	XV_COMBINE	0

	Use only AOS/LOS	XV_AOS_LOS	1
	Use only physical mask	XV_PHYSICAL	2
	Use mask from file	XV_FROM_FILE	3

7.4.4 Output parameters

Table 18: Output parameters of xv_station_vis_time function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_station_vis_time	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long		Number of visibility segments returned to the user		≥ 0
bgn_orbit	long*	all	Orbit number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		> 0
bgn_second	long*	all	Seconds since ascending node, begin of visibility segment i bgn_second[i-1], i = 1, number_segments	s	≥0 < orbital period
bgn_microsec	long*	all	Micro seconds within second begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments	µs	≥0 ≤ 999999
bgn_cycle	long*	all	Cycle number, begin of visibility segment i bgn_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
end_orbit	long*	all	Orbit number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		> 0
end_second	long*	all	Seconds since ascending node, end of visibility segment i end_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period
end_microsec	long*	all	Micro seconds within second	µs	≥0

			end of visibility segment i end_microsec[i-1], i = 1, number_segments		≤ 999999
end_cycle	long*	all	Cycle number, end of visibility segment i end_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
zdop_orbit	long*	all	Orbit number, time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_orbit[i-1], i = 1, number_segments		> 0
zdop_second	long*	all	Seconds since ascending node, time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_second[i-1], i = 1, number_segments	s	>= 0 < orbital period
zdop_microsec	long*	all	Micro seconds within second time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_microsec[i-1], i = 1, number_segments	µs	0 =< =< 999999
zdop_cycle	long*	all	Cycle number, time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_second[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
ierr[XV_NUM_ERR_STATION_VIS_TIME]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_station_vis_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.4.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_station_vis_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_station_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 19: Error messages and codes for xv_station_vis_time

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error in input parameter Orbit Id.	Computation not performed	XV_CFI_STATION_VIS_TIME_ORBIT_STATUS_ERR	0
ERR	Error in input parameter to stavistime.	Computation not performed	XV_CFI_STATION_VIS_TIME_INPUTS_CHECK_ERR	1
ERR	Input parameter "orbit_type" is out of range.	Computation not performed	XV_CFI_STATION_VIS_TIME_ORBIT_TYPE_ERR	2
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_STATION_VIS_TIME_ORBIT_MODEL_ERR	3
ERR	Error transforming start orbit from relative to absolute orbits.	Computation not performed	XV_CFI_STATION_VIS_TIME_REL_TO_ABS_START_ERR	4
ERR	Error transforming stop orbit from relative to absolute orbits	Computation not performed	XV_CFI_STATION_VIS_TIME_REL_TO_ABS_STOP_ERR	5
ERR	Error reading the Orbit scenario file.	Computation not performed	XV_CFI_STATION_VIS_TIME_OSF_READ_E RR	6
ERR	Error reading the swath template file.	Computation not performed	XV_CFI_STATION_VIS_TIME_SWATH_FLAG_ERR	7
ERR	Error reading the swath template file.	Computation not performed	XV_CFI_STATION_VIS_TIME_SWATH_READ_ERR	8
ERR	Error wrong swath type selected.	Computation not performed	XV_CFI_STATION_VIS_TIME_SWATH_TYPE_ERR	9

ERR	Swath file is not compatible with the orbit file	Computation not performed	XV_CFI_STATION_VIS_TIME_WRONG_SWATH_ERR	10
WAR N	Warning, start orbit is outside range of OSF.	Computation performed. Message to inform the user.	XV_CFI_STATION_VIS_TIME_FIRST_ORBIT_WARN	11
WAR N	Warning, stop orbit is outside range of OSF.	Computation performed. Message to inform the user.	XV_CFI_STATION_VIS_TIME_LAST_ORBIT_WARN	12
ERR	Actual stop orbit is earlier than actual start orbit.	Computation not performed	XV_CFI_STATION_VIS_TIME_WRONG_INTERVAL_ERR	13
ERR	Error obtaining orbital information in orbit info.	Computation not performed	XV_CFI_STATION_VIS_TIME_ORBIT_INFO_ERR	14
WAR N	Warning, there is an orbital change within the requested orbits.	Computation performed. Message to inform the user.	XV_CFI_STATION_VIS_TIME_ORBIT_CHANGE_WARN	15
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_STATION_VIS_TIME_INTERNAL_MEMORY_ERR	16
ERR	There is a potential memory overload, try with a smaller orbital interval.	Computation not performed	XV_CFI_STATION_VIS_TIME_POTENTIAL_MEMORY_ERR	17
ERR	Orbital information does not coincide with reference swath.	Computation not performed	XV_CFI_STATION_VIS_TIME_INCONSISTENT_SWATH_ERR	18
ERR	Error read info the ground station's mask data file.	Computation not performed	XV_CFI_STATION_VIS_TIME_READ_STA_E RR	19
ERR	Error transforming the station's mask into an equivalent zone.	Computation not performed	XV_CFI_STATION_VIS_TIME_AZEL2LONLAT_ERR	20
ERR	Error calling ZONEVISTIME to calculate transitions.	Computation not performed	XV_CFI_STATION_VIS_TIME_ZONE_VIS_TIME_CALL_ERR	21
ERR	Error refining intersection time.	Computation not performed	XV_CFI_STATION_VIS_TIME_CALL_STAVIS_ERR	22
WAR N	Accuracy of 0.001 deg in elevation not reached in orbit %li. Orbit too close to the mask limit.	Computation performed. Message to inform the user.	XV_CFI_STATION_VIS_TIME_CALL_STAVIS_WARN	23
ERR	Error allocating memory for the time segments.	Computation not performed.	XV_CFI_STATION_VIS_TIME_SEGMENTS_MEMORY_ERR	24
ERR	Error calculating zero	Computation not performed	XV_CFI_STATION_VI	25

	doppler interval.		S_TIME_ZERO_DOPPL ER_ERR	
WARN	Segment longer than half nodal period deleted.	Computation performed. Message to inform the user.	XV_CFI_STATION_VI S_TIME_LONG_SEGM _SKIPPED_WARN	26
ERR	Error transforming from absolute to relative.	Computation not performed	XV_CFI_STATION_VI S_TIME_ABS_TO_REL _ERR	27
ERR	Error in the mask type read from the mask data given in the file	Computation not perfomed	XV_CFI_STATION_VIS_TIM E_MASK_FROM_FILE_MA SK_TYPE_ERR	32
ERR	Error finding the spacecraft for the station when mask data given from file	Computation not performed	XV_CFI_STATION_VIS_TIM E_MASK_FROM_FILE_NO _SC_ERR	33
ERR	Error converting zone point array to zone record	Computation not perfomed	XV_CFI_STATION_VIS_TIM E_CONVERT_ZONE_ERR	34
ERR	Input orbit interval is completely outside STF validity interval	Computation not perfomed	XV_CFI_STATION_VIS_TIM E_ORBIT_INTERVAL_STF_ ERR	35
WARN	Input orbit interval is partially outside STF validity interval	Computation perfomed	XV_CFI_STATION_VIS_TIM E_ORBIT_INTERVAL_STF_ WARN	36
ERR	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation not perfomed	XV_CFI_STATION_VIS_TIM E_OSF_NON_LIN_STF_OL D_WARN	37
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation perfomed	XV_CFI_STATION_VIS_TIM E_SWATH_FLAG_LARGER _THAN_LIN_APPROX_VAL _WARN	38
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_STATION_VIS_TIM E_GEO_SAT_ERR	39
WARN	Deprecated function. Use xv_stationvistime_compute instead	Computation performed	XV_CFI_STATION_VIS_TIM E_DEPRECATED_WARN	40
WARN	Visibility computations with precise propagator can be very slow	Computation performed	XV_CFI_STATION_VIS_TIM E_PRECISE_PROPAG_WA RN	41

7.5 xv_station_vis_time_no_file

7.5.1 Overview

Note: this function is deprecated. Use **xv_stationvistime_compute** instead.

The **xv_station_vis_time_no_file** function computes ground station visibility segments, the orbital segments for which the satellite is visible from a ground station located at the surface of the Earth.

The aim of this function is to provide another interface for the function **xv_station_vis_time** in which the station and the swath are not provided with files but with data structures (see section 7.2.2).

7.5.2 Calling interface

For C programs, the call to **xv_station_vis_time_no_file** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
      xo_orbit_id orbit_id = {NULL};
      long     swath_flag, orbit_type,
               start_orbit, start_cycle,
               stop_orbit, stop_cycle,
               mask, number_segments,
               *bgn_orbit, *bgn_second,
               *bgn_microsec, *bgn_cycle,
               *end_orbit, *end_second,
               *end_microsec, *end_cycle,
               *zdop_orbit, *zdop_second,
               *zdop_microsec, *zdop_cycle,
               ierr[XV_NUM_ERR_STATION_VIS_TIME],
               status;
      double   aos_elevation, los_elevation, min_duration;
      xd_stf_file  stf_data;
      xd_station_rec station_data;

      status = xv_station_vis_time_no_file(
                       &orbit_id, &orbit_type,
                       &start_orbit, &start_cycle,
                       &stop_orbit, &stop_cycle,
                       &stf_data, &station_data,
                       &mask, &aos_elevation, &los_elevation,
                       &min_duration,
                       &number_segments,
                       &bgn_orbit, &bgn_second,
```

```
                          &bgn_microsec, &bgn_cycle,
                          &end_orbit, &end_second,
                          &end_microsec, &end_cycle,
                         &zdop_orbit, &zdop_second,
                         &zdop_microsec, &zdop_cycle,
                          ierr);


        /* Or, using the run_id */
        long run_id;

        status = xv_station_vis_time_no_file_run(
                          &run_id, &orbit_type,
                          &start_orbit, &start_cycle,
                          &stop_orbit, &stop_cycle,
                          &stf_data, &station_data,
                          &mask, &aos_elevation, &los_elevation,
                          &min_duration,
                          &number_segments,
                          &bgn_orbit, &bgn_second,
                          &bgn_microsec, &bgn_cycle,
                          &end_orbit, &end_second,
                          &end_microsec, &end_cycle,
                         &zdop_orbit, &zdop_second,
                         &zdop_microsec, &zdop_cycle,
                          ierr);
}
```

7.5.3 Input parameters

Table 20: Input parameters of xv_station_vis_time_no_file

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤ repeat cycle
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle) If it is not initialized with orbital changes, stop orbit will be set to the last orbit in orbit_id initialization.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤ repeat cycle
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
stf_data	xd_stf	-	Swath template data (structure	-	-

	_file		described in [D_H_SUM]). The swath structure can be got by: • Reading a swath template file with the CFI function **xd_read_stf**. • Generating the swath data with the CFI function **xv_gen_swath_no_file**		
station_data	xd_station_rec	-	Station data (structure described in [D_H_SUM]) that can be got by reading a station from a station database file with the CFI function **xd_read_station**.	-	-
mask	long		mask used to define visibility = XV_COMBINE combine AOS/LOS elevations and physical mask (nominal mode) = XV_AOS_LOS consider only AOS/LOS elevations = XV_PHYSICAL consider only physical mask = XV_FROM_FILE consider mask given in the Station Database File		all
aos_elevation	double		Minimum elevation to consider at AOS (i.e. before considering start of visibility). Not used if mask=XV_FROM_FILE	deg	≥ 0.0
los_elevation	double		Maximum elevation to consider at LOS (i.e. before considering end of visibility). Not used if mask=XV_FROM_FILE	deg	≥0.0 ≤ aos_elevation
min_duration	double		Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0.0

It is also possible to use enumeration values rather than integer values for some of the input arguments, as shown in the table below:

Input	Description	Enumeration value	long
mask	Combine AOS/LOS and physical mask	XV_COMBINE	0
	Use only AOS/LOS	XV_AOS_LOS	1
	Use only physical mask	XV_PHYSICAL	2

	Use mask from file	XV_FROM_FILE	3

7.5.4 Output parameters

Table 21: Output parameters of xv_station_vis_time_no_file function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_station_vis_time_no_file	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long		Number of visibility segments returned to the user		≥ 0
bgn_orbit	long*	all	Orbit number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		> 0
bgn_second	long*	all	Seconds since ascending node, begin of visibility segment i bgn_second[i-1], i = 1, number_segments	s	≥0 < orbital period
bgn_microsec	long*	all	Micro seconds within second begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments	µs	≥0 ≤999999
bgn_cycle	long*	all	Cycle number, begin of visibility segment i bgn_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
end_orbit	long*	all	Orbit number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		> 0
end_second	long*	all	Seconds since ascending node, end of visibility segment i end_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period
end_microsec	long*	all	Micro seconds within second	µs	≥0

			end of visibility segment i end_microsec[i-1], i = 1, number_segments		≤ 999999
end_cycle	long*	all	Cycle number, end of visibility segment i end_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
zdop_orbit	long*	all	Orbit number, time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_orbit[i-1], i = 1, number_segments		> 0
zdop_second	long*	all	Seconds since ascending node, time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_second[i-1], i = 1, number_segments	s	>= 0 < orbital period
zdop_microsec	long*	all	Micro seconds within second time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_microsec[i-1], i = 1, number_segments	µs	0 =< =< 999999
zdop_cycle	long*	all	Cycle number, time of zero doppler (-1 if no zero doppler within corresponding visibility segment) zdop_second[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
ierr[XV_NUM_ER R_STATION_VIS_ TIME]	long		Error status flags		

<u>Memory Management:</u> Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_station_vis_time_no_file** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.5.5 Warnings and errors

The error and warning messages and codes for **xv_station_vis_time_no_file** are the same than for **xv_station_vis_time** (see Table 19) .

The error messages/codes can be returned by the CFI function **xv_get_msg/xv_get_code** after translating the returned status vector into the equivalent list of error messages/codes. The function identifier to be used in that functions is XV_STATION_VIS_TIME_ID (from Table 2).

7.6 xv_stationvistime_compute

7.6.1 Overview

The **xv_stationvistime_compute** function computes ground station visibility segments, the orbital segments for which the satellite is visible from one or several ground stations located at the surface of the Earth.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as seconds elapsed since the ascending node crossing.

If more than one ground station is provided, the visibility segments are computed internally for each of them. Those segments are merged and ordered by start time. In the output visibility segments, it is listed the stations from which the satellite is visible in each segment.

In addition, **xv_stationvistime_compute** calculates for every visibility segment the time of zero-doppler (i.e. the time at which the range-rate to the station is zero). It is computed per station, is case several stations are used as input.

xv_stationvistime_compute requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data and files, also for precise propagation if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**).
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
  - TLE files

    Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.
- The swath_id (xv_swath_id, initialized using xv_swath_id_init -section 7.32-), which provides the Instrument Swath data, describing the area seen by the relevant instrument all along the current orbit.
- The Station data (xv_station_info_list), describing the location and the physical mask of each ground station, and the mask parameters for a list of spacecrafts from each station (considered only when mask 'from file' option is selected).

The time intervals (xv_time_interval) used by **xv_stationvistime_compute** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times. Moreover, the segments will be ordered chronologically.

NOTE: If **xv_stationvistime_compute** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behavior depends on the type of the data used to initialize the swath_id (via xv_swath_id_init, section 7.32):

•If a **swath template file** is used, **xv_stationvistime_compute** automatically will ignore the orbits that do not correspond with the template file (i.e. no visibility segments will be generated for those orbits), since swath template file is generated from a reference orbit with a particular geometry, so it is not valid for a different geometry.

•If a **swath definition file** is introduced, **xv_stationvistime_compute** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath template files valid for every orbital change are generated to perform the calculations.

NOTE 2:If a swath template file with the variable header tags *Start_Validity_Range* and *Stop_Validity_Range* is used as input, only the segments belonging to that orbit range will be returned.

NOTE 3: If a swath definition file is introduced, it can be also introduced every how many orbits the swath template file must be recomputed (according to the number of regeneration orbits used in swath id initialization). If the orbit_id has been initialized with an OSF file with MLST non linear terms and the number of regeneration orbits is greater than the linear approximation validity, the recomputation of swath template file will be done every linear approximation validity orbits.

7.6.2 Usage Hints

7.6.2.1 Use of input xp_attitude_def struct

The definition of the input structure xp_attitude_def can be consulted in section 6.3 of [POINT_SUM].

The “type” field defines how this struct is used, and it can take the following values:

- XP_NONE_ATTITUDE: no attitude defined in struct. In this case, when the Swath Template File must be computed internally, the attitudes defined in Swath Definition File are used.
- XP_SAT_NOMINAL_ATT, XP_SAT_ATT, XP_INSTR_ATT: the attitudes defined in the structure are used in internal Swath Template File generation. The “type” field in this case indicates the target frame for the computation.

7.6.2.2 Use of input xv_station_info_list struct

The station or stations to be used in algorithm are passed to xv_stationvistime_compute function with the struct *xv_station_info_list* (see section 6.3 for description). It contains the following fields:

- *calc_flag*: it can take the enumeration values XV_COMPUTE or XV_DO_NOT_COMPUTE. This flag indicates if the extra information regarding the stations (zero doppler time) must be computed or not.
- *num_rec*: Indicates the number of input stations where the visibility is going to be computed.
- *station_info*: description of the stations to be computed. Every position of the array is a *xv_station_info* struct . The value of *type* field indicates the type of station data:

   - If *type* is equal to enum value XV_USE_STATION_FILE, then the station *station_id* is read from *station_db_filename* station file. If AOS, LOS and mask are not defined in the file, the values are taken from the fields *default_aos*, *default_los* and *default_mask.*
   - If *type* is equal to enum value XV_USE_STATION_FILE_AND_MASK_OVERRIDE, then the station *station_id* is read from *station_db_filename* station file. In this case, the values used in computations for AOS, LOS and mask are taken from the fields *default_aos*, *default_los* and *default_mask*, not from the information read from station file.
   - If *type* is equal to enum value XV_USE_STATION_DATA, then the information contained in *station_data* field is used (see [D_H_SUM] for description of *xd_station_rec*). If AOS, LOS and mask are not defined in the struct, the values are taken from the fields *default_aos*, *default_los* and *default_mask.*
   - If *type* is equal to enum value XV_USE_STATION_DATA_AND_MASK_OVERRIDE, then the information contained in *station_data* field is used (see [D_H_SUM] for description of *xd_station_rec*). In this case, the values used in computations for AOS, LOS and mask are taken from the fields *default_aos*, *default_los* and *default_mask*, not from the information read from station file.

- *min_duration*: indicates the minimum duration for the segments (seconds).

7.6.3 Calling interface

For C programs, the call to **xv_stationvistime_compute** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
        xo_orbit_id orbit_id = {NULL};
        xp_attitude_def attitude_def;
        xv_swath_id swath_id = {NULL};
        xv_station_info_list station_info_list;
        xv_time_interval search_interval;
        xv_stationvisibility_interval_list visibility_interval_list;
        long ierr[XV_NUM_ERR_STATIONVISTIME_COMPUTE];

        status = xv_stationvistime_compute(
                          &orbit_id, &attitude_def,
                          &swath_id, &station_info_list,
                          &search_interval,
                          &visibility_interval_list, ierr);
```

7.6.4 Input parameters

Table 22: Input parameters of xv_stationvistime_compute

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
attitude_def	xp_attitude_def*	-	Structure with the definition of the attitudes.	-	-
swath_id	xv_swath_id*	-	Swath id.	-	-
station_info_list	xv_station_info_list*	-	List of station where the visibility is going to be computed.	-	-

It is also possible to use enumeration values rather than integer values for some of the input arguments, as shown in the table below:

Input	Description	Enumeration value	long
mask	Combine AOS/LOS and physical mask	XV_COMBINE	0
	Use only AOS/LOS	XV_AOS_LOS	1
	Use only physical mask	XV_PHYSICAL	2
	Use mask from file	XV_FROM_FILE	3

7.6.5 Output parameters

Table 23: Output parameters of xv_stationvistime_compute function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_station_vis_time	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
visibility_interval_list	xv_stationvisibility_interval_list*	-	List of visibility segments and additional information (zero doppler and station names)	-	-
ierr[XV_NUM_ERR_STATIONVISTIME_COMPUTE]	long		Error status flags		

Memory Management: Note that the output visibility segment list (xv_stationvisibility_interval_list->visibility_interval) is a pointer to the list of segments computed inside xv_stationvistime_compute. The memory for this dynamic array is allocated within the **xv_stationvistime_compute** function. So the user will only have to declare the variable xv_stationvisibility_interval_list. However, once the function has returned without error, the user will have the responsibility of freeing the memory for the pointers inside that variable and the rest of structs inside xv_station_coverage_info_list struct.

7.6.6 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_stationvistime_compute** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_stationvistime_compute** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 24: Error messages and codes for xv_stationvistime_compute

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Input parameter \"Orbit Id\" is wrong.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_STATUS_ERR	0
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_GEO_SAT_ERR	1
ERR	Swath id not initialized	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_SWATH_STATUS_ERR	2
ERR	Error in input parameter to stavistime.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_INPUTS_CHECK_ERR	3
ERR	Input parameter \"orbit_type\" is out of range.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_TYPE_ERR	4
ERR	Input parameter \"swath_flag\" is out of range.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_SWATH_FLAG_ERR	5
ERR	Warning, start orbit is outside range of OSF.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_FIRST_ORBIT_WARN	6
ERR	Error transforming start orbit from relative to absolute orbits.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_REL_TO_ABS_START_ERR	7
ERR	Actual stop orbit is earlier than actual start orbit.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_WRONG_INTERVAL_ERR	8
ERR	Swath file is not compatible with the orbit file	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_WRONG_SWATH_ERR	9

ERR	Error wrong swath type selected.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_SWATH_TYPE_ERR	10
ERR	Error obtaining orbital information in orbit info .	Computation not performed.	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_INFO_ERR	11
ERR	Orbital information does not coincide with reference swath.	Computation not performed.	XV_CFI_STATIONVISTIME_COMPUTE_INCONSISTENT_SWATH_ERR	12
ERR	Warning, there is an orbital change within the requested orbits .	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_CHANGE_WARN	13
ERR	Error computing segments for one station	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_STA_COMPUTE_ERR	14
ERR	Error allocating memory for the time segments.	Computation not performed.	XV_CFI_STATIONVISTIME_COMPUTE_SEGMENTS_MEMORY_ERR	15
ERR	Error transforming from absolute to relative.	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_ABS_TO_REL_ERR	16
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_MODEL_ERR	17
ERR	Error reading Swath File: %s	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_READ_SWATH_FILE_ERR	18
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_INTERVAL_STF_ERR	19
WARN	Input orbit interval is partially outside STF validity interval	Computation performed. Message to inform the user.	XV_CFI_STATIONVISTIME_COMPUTE_ORBIT_INTERVAL_STF_WARN	20
WARN	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation performed. Message to inform the user.	XV_CFI_STATIONVISTIME_COMPUTE_OSF_NON_LIN_STF_OLD_WARN	21
ERR	Error computing overlap for multistations	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_OVERLAP_ERR	22
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation performed. Message to inform the user.	XV_CFI_STATIONVISTIME_COMPUTE_SWATH_FLAG_LARGER_THAN_LIN_APPROX_VAL_WARN	23
ERR	Error transforming time to orbit	Computation not performed.	XV_CFI_STATIONVISTIME_COMPUTE_TIME_TO_ORBIT_ERR	24

ERR	Error checking output segments	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_CHECK_SEGMENTS_ERR	25
ERR	Error computing UTC time	Computation not performed.	XV_CFI_STATIONVISTIME_COMPUTE_GET_UTC_TIME_ERR	26
ERR	Input file is not a swath file	Computation not performed	XV_CFI_STATIONVISTIME_COMPUTE_DETECT_SWATH_TYPE_ERR	32
WARN	Visibility computations with precise propagator can be very slow	Computation performed	XV_CFI_STATIONVISTIME_COMPUTE_PRECISE_PROPAG_WARN	33

7.7 xv_sc_vis_time

7.7.1 Overview

The **xv_sc_vis_time** function computes all the orbital segments for which Communication Terminal of a target satellite (LEO or GEO) is visible from Communication Terminal of a source satellite (LEO).

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as seconds elapsed since the ascending node crossing.

xv_sc_vis_time requires access to the orbit_id (xo_orbit_id) data structure of both satellites. For orbit id initialization please refer to [ORBIT_SUM].

Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.

The time intervals used by **xv_sc_vis_time** are expressed in absolute or relative orbit numbers (with respect to source satellite). This is valid for both:

- input parameter “Orbit Range”: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle number should be used, otherwise, this the cycle number will be a dummy parameter.
- output parameter “Target Satellite Visibility Segments”: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

Users who need to use processing times must make use of the conversion routines provided in [ORBIT_SUM] (**xo_time_to_orbit** and **xv_orbit_to_time** functions).

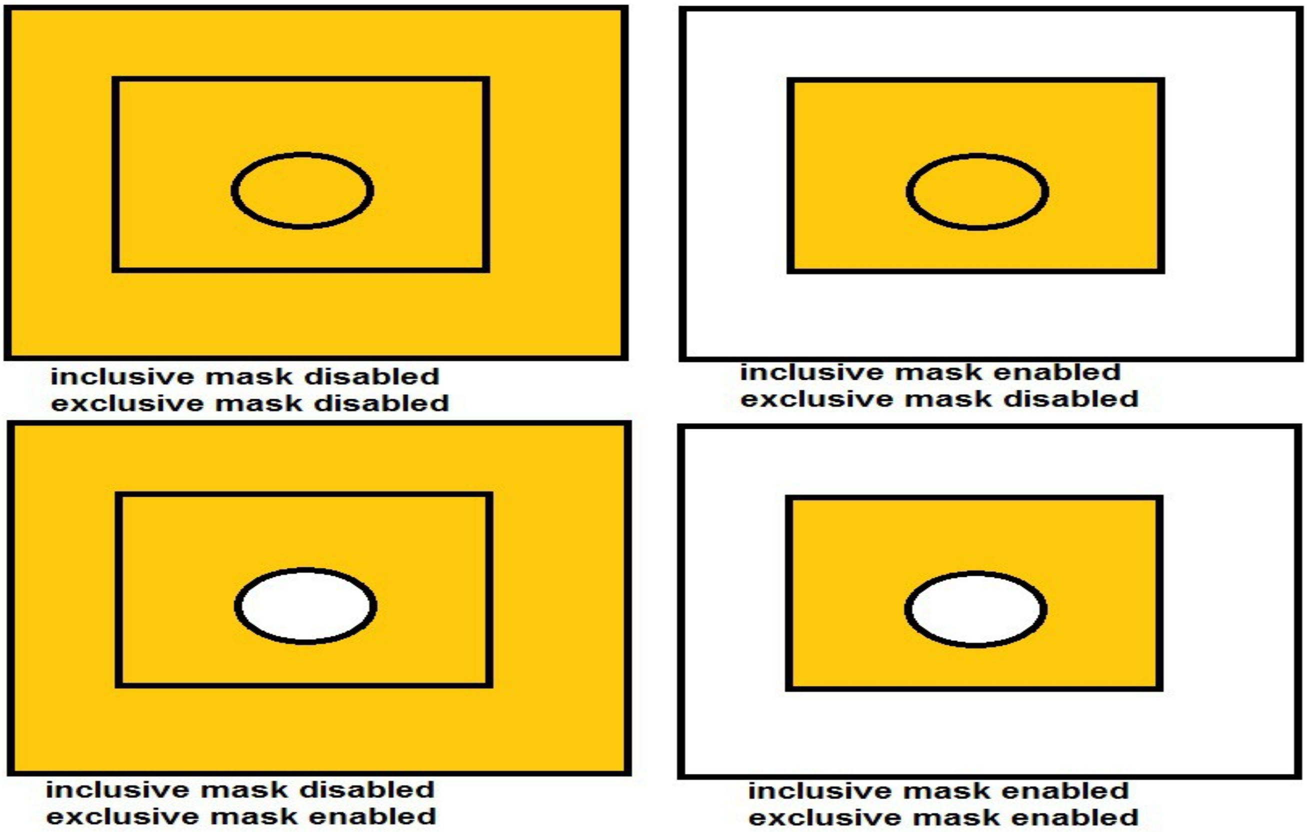
The **xv_sc_vis_time** function considers the following sources of occultation, which can be configured through xv_link_data input struct:

- Earth plus a minimum tangent height.
- Satellite inclusive and exclusive masks, which are zones of azimuth and elevation where visibility is possible (inclusive mask) or not possible (exclusive mask). These masks can be used to model constraints (e.g. mechanical) or occlusion of the field of view.

In the following figure, the behaviour of the mask is explained. The four mask combinations are represented:

- Inclusive mask **disabled**, exclusive mask **disabled**.
- Inclusive mask **enabled**, exclusive mask **disabled**.
- Inclusive mask **disabled**, exclusive mask **enabled**.
- Inclusive mask **enabled**, exclusive mask **enabled**.

In every case, the full field of view is represented. The internal rectangle represents the inclusive mask and the internal ellipse represents the exclusive mask. In every case, the zone in orange colour background is the field of view allowed by the enabled masks.



Notes about definition of masks:

- The masks are defined as closed zones.
- These zones are defined in the input struct xv_link_data with arrays of azimuth and elevation points that define a polygon in the azimuth-elevation plane (last point in array is closed with first point in array internally).
- It must be distinguished between azimuth = 0. deg and azimuth = 360. deg, since they are considered different in the azimuth-elevation plane; this has been done to make the definition of masks easier.
- The masks are enabled or disabled using the field status of xv_az_el_mask struct, setting its value to XP_TRUE or XP_FALSE respectively.

7.7.2 Calling interface

For C programs, the call to **xv_sc_vis_time** is (<u>input</u> parameters are <u>underlined</u>):

```
#include"explorer_visibility.h"
{
        xo_orbit_id orbit_id1 = {NULL};
        xo_orbit_id orbit_id2 = {NULL};
        xp_sat_nom_trans_id sat_nom_trans_id1 = {NULL};
```

```
xp_sat_nom_trans_id sat_nom_trans_id2 = {NULL};
xp_sat_trans_id     sat_trans_id1 = {NULL};
xp_sat_trans_id     sat_trans_id2 = {NULL};
xp_instr_trans_id   instr_trans_id1 = {NULL};
xp_instr_trans_id   instr_trans_id2 = {NULL};
long          orbit_type,
              start_orbit, start_cycle,
              stop_orbit, stop_cycle,
              number_segments,
              *bgn_orbit, *bgn_second,
              *bgn_microsec, *bgn_cycle,
              *end_orbit, *end_second,
              *end_microsec, *end_cycle,
              ierr[XV_NUM_ERR_SC_VIS_TIME],
              status;
double        min_duration;
xv_link_data link_data;

status = xv_sc_vis_time(
                 &orbit_id1, &sat_nom_trans_id1,
                 &sat_trans_id1, &instr_trans_id1, &orbit_type,
                 &start_orbit, &start_cycle,
                 &stop_orbit, &stop_cycle,
                 &orbit_id2, &sat_nom_trans_id2,
                 &sat_trans_id2, &instr_trans_id2,
                 &link_data, &min_duration,
                 &number_segments,
                 &bgn_orbit, &bgn_second,
                 &bgn_microsec, &bgn_cycle,
                 &end_orbit, &end_second,
                 &end_microsec, &end_cycle,
                 ierr);

/* Or, using the run_id */
long run_id1, run_id2;

status = xv_sc_vis_time_run(
                 &run_id1, &run_id2, &orbit_type,
                 &start_orbit, &start_cycle,
                 &stop_orbit, &stop_cycle,
                 &link_data, &min_duration,
                 &number_segments,
```

```
                                &bgn_orbit, &bgn_second,
                                &bgn_microsec, &bgn_cycle,
                                &end_orbit, &end_second,
                                &end_microsec, &end_cycle,
                                ierr);
    }
```

7.7.3 Input parameters

Table 25: Input parameters of xv_sc_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id1	xo_orbit_id*	-	Structure that contains the orbit data of the source satellite	-	-
sat_nom_trans_id1	xp_sat_nom_trans_id*	-	Structure that contains the Sat. Nom.. Trans. of the source satellite	-	-
sat_trans_id1	xp_sat_trans_id*	-	Structure that contains the Sat. Trans. of the source satellite	-	-
instr_trans_id1	xp_instr_trans_id*	-	Structure that contains the Instr. Trans. of the source satellite	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤ repeat cycle
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≤ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤ repeat cycle

			complete cycle)		
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or ≤first cycle in osf
orbit_id2	xo_orbit_id*	-	Structure that contains the orbit data of the target satellite	-	-
sat_nom_trans_id2	xp_sat_nom_trans_id*	-	Structure that contains the Sat. Nom.. Trans. of the source satellite	-	-
sat_trans_id2	xp_sat_trans_id*	-	Structure that contains the Sat. Trans. of the target satellite	-	-
instr_trans_id2	xp_instr_trans_id*	-	Structure that contains the Instr. Trans. of the target satellite	-	-
link_data	xv_link_data*	-	Link data (minimum tangent height and masks)	[m] for minimum height [deg] for azimuth and elevation	Height ≥ 0. Azimuth range: [0., 360.] Elevation range: [-90., 90.]
min_duration	double	-	Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥0.0

7.7.4 Output parameters

Table 26: Output parameters of xv_sc_vis_time function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_sc_vis_time	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long		Number of visibility segments returned to the user		≥0
bgn_orbit	long*	all	Orbit number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		> 0
bgn_second	long*	all	Seconds since ascending node, begin of visibility segment i bgn_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period
bgn_microsec	long*	all	Micro seconds within second begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments	ms	≥ 0 ≤ 999999
bgn_cycle	long*	all	Cycle number, begin of visibility segment i bgn_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
end_orbit	long*	all	Orbit number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		> 0
end_second	long*	all	Seconds since ascending node, end of visibility segment i end_second[i-1], i = 1, number_segments	s	≥ 0 <orbital period
end_microsec	long*	all	Micro seconds within second	ms	≥ 0

			end of visibility segment i end_microsec[i-1], i = 1, number_segments		≤ 999999
end_cycle	long*	all	Cycle number, begin of visibility segment i bgn_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
ierr[XV_NUM_E RR_SC_VIS_TI ME]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_sc_vis_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.7.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_sc_vis_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_sc_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 27: Error messages of xv_sc_vis_time

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	No computation performed	XV_CFI_SC_VIS_TIME_INTERNAL_MEMORY_ERR	0
ERR	Wrong input orbit Id.	No computation performed	XV_CFI_SC_VIS_TIME_ORBIT_STATUS_ERR	1
ERR	Error in input parameters.	No computation performed	XV_CFI_SC_VIS_TIME_XV_SC_INPUTS_CHECK_ERR	2
ERR	Input parameter \"orbit_type\" is out of range.	No computation performed	XV_CFI_SC_VIS_TIME_ORBIT_TYPE_ERR	3
WARN	Input \"start_orbit\" below first OSF orbit: take first OSF orbit for computations.	Computation performed	XV_CFI_SC_VIS_TIME_START_ORBIT_WARN	4
ERR	Error in absolute start orbit computation.	No computation performed	XV_CFI_SC_VIS_TIME_REL_TO_ABS_START_ERR	5
ERR	Error in absolute stop orbit computation.	No computation performed	XV_CFI_SC_VIS_TIME_REL_TO_ABS_STOP_ERR	6
ERR	Wrong orbit range.	No computation performed	XV_CFI_SC_VIS_TIME_WRONG_ORBIT_RANGE_ERR	7
ERR	Error in orbit parameters computation. Orbit no: (%ld).	No computation performed	XV_CFI_SC_VIS_TIME_XV_ORBIT_INFO_ERR	8
ERR	Error performing a time transformation.	No computation performed	XV_CFI_SC_VIS_TIME_TIME_CHANGE_ERR	9
ERR	Error checking the visibility.	No computation performed	XV_CFI_SC_VIS_TIME_VIS_CHECK_ERR	10
WARN	First orbit starts with visibility.	Computation performed	XV_CFI_SC_VIS_TIME_FIRST_ORBIT_VIS_WARN	11

ERR	Maximum number of iterations. Orbit no: (%ld).	No computation performed	XV_CFI_SC_VIS_TIME_MAX_NUMBER_ITER_ERR	12
ERR	Error in time computations. Orbit no: (%ld).	No computation performed	XV_CFI_SC_VIS_TIME_XV_TIME_SEC_ERR	13
ERR	Error transforming absolute to relative begin segments.	No computation performed	XV_CFI_SC_VIS_TIME_ABS_TO_REL_BGN_ERR	14
ERR	Error transforming absolute to relative end segments.	No computation performed	XV_CFI_SC_VIS_TIME_ABS_TO_REL_END_ERR	15
WARN	Last orbit ends with visibility	Computation performed	XV_CFI_SC_VIS_TIME_LAST_ORBIT_VIS_WARN	16
ERR	Wrong satellite nominal attitude Id.	No computation performed	XV_CFI_SC_VIS_TIME_SAT_NOM_ATT_STATUS_ERR,	17
ERR	Wrong satellite attitude Id.	No computation performed	XV_CFI_SC_VIS_TIME_SAT_ATT_STATUS_ERR	18
ERR	Wrong instrument attitude Id.	No computation performed	XV_CFI_SC_VIS_TIME_INSTR_ATT_STATUS_ERR	19
WARN	Visibility computations with precise propagator can be very slow	Computation performed	XV_CFI_SC_VIS_TIME_PRECISE_PROPAG_WARN	20

7.8 xv_swath_pos

7.8.1 Overview

Note: this function is deprecated. Use **xv_swathpos_compute** instead.

The **xv_swath_pos** function computes the location of a swath at a given time.

Swath location is expressed as[1]:

- longitude
- latitude
- altitude

for n points (with n >=1). In Figure 2 we can see an example.

xv_swath_pos requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data and files, also with precise propagation if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)

1 For inertial swaths, right ascension and declination are used instead of longitude and latitude

- Restituted orbit files
- DORIS Preliminary orbit files
- DORIS Navigator files
- TLE file

- the Instrument Swath data, describing the area seen by the relevant instrument all along the current orbit. The swath file is produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function) and the data structure can be got by reading the file with **xd_read_stf**.

The input time used by **xv_swath_pos** is expressed in orbit-relative time.

Users who need to use processing time must make use of the conversion routine provided in EO_VISIBILITY (**xv_time_to_orbit** and **xv_orbit_to_time** functions).

NOTE: Since the swath template file is generated from a reference orbit, it is not allowed to use **xv_swath_pos** for an orbit in the orbit scenario file with different repeat cycle or cycle length. If this would happen, **xv_swath_pos** will return an error an no computation will be performed.

7.8.2 Calling sequence of xv_swath_pos

For C programs, the call to **xv_swath_pos** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
      xo_orbit_id  orbit_id = {NULL};
      long         orbit_type,
                   orbit, second, microsec, cycle,
                   ierr[XV_NUM_ERR_SWATH_POS], status;
      double       *longitude, *latitude, *altitude;
      xd_stf_file  stf_data;


      status = xv_swath_pos(&orbit_id,
                            &stf_data,
                            &orbit_type,
                            &orbit, &second, &microsec, &cycle,
                            longitude, latitude, altitude,
                            ierr);

      /* Or, using the run_id */
      long run_id;

      status = xv_swath_pos_run(&run_id,
                                &stf_data,
                                &orbit_type,
                                &orbit, &second, &microsec, &cycle,
                                longitude, latitude, altitude,
                                ierr);
}
```

7.8.3 Input parameters xv_swath_pos

Table 28: Input parameters of xv_swath_pos

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
stf_data	xd_stf_file		Swath Template data structure	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete
orbit	long		Orbit number		> 0
second	long		Seconds since ascending node	s	>= 0 < orbital period
microsec	long		Micro seconds within second	ms	0 =< =< 999999
cycle	long		Cycle number		>0

7.8.4 Output parameters xv_swath_pos

Table 29: Output parameters of xv_swath_pos

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_swath_pos	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
longitude	double*	all	longitude (right ascension for inertial swaths) of points of the swath. The user must reserve as many array positions as the number of points of the instantaneous swath.	deg	[-180, 180]
latitude	double*	all	latitude (declination for inertial swaths) of points of the swath.	deg	[-90, 90]

			The user must reserve as many array positions as the number of points of the instantaneous swath.		
altitude	double*	all	altitude of point is of the swath. The user must reserve as many array positions as the number of points of the instantaneous swath.	m	
ierr[XV_NUM_ERR_SWATH_POS]	long		Error status flags		

7.8.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_swath_pos** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_swath_pos** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 30: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong orbit Id.	Computation not performed	XV_CFI_SWATH_POS _ORBIT_STATUS_ERR	0
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_SWATH_POS _ORBIT_MODEL_ERR	1
ERR	Orbital information does not coincide with reference swath.	Computation not performed	XV_CFI_SWATH_POS _INCONSISTENT_SW ATH_ERR	2
ERR	Orbit number must be positive.	Computation not performed	XV_CFI_SWATH_POS _ORB_NUM_LIM_ERR	3
ERR	Seconds since ascending node must be zero or positive.	Computation not performed	XV_CFI_SWATH_POS _SEC_LIM_ERR	4
ERR	MicroSeconds must be zero or positive	Computation not performed	XV_CFI_SWATH_POS _MICROSEC_1ST_ERR	5
ERR	MicroSeconds can not be bigger than 999999.	Computation not performed	XV_CFI_SWATH_POS _MICROSEC_2ND_ER R	6
ERR	Orbit type switch out of range.	Computation not performed	XV_CFI_SWATH_POS _ORBIT_TYPE_ERR	7
ERR	Cycle number must be positive.	Computation not performed	XV_CFI_SWATH_POS _CYCLE_ERR	8
ERR	Orbit number is not included in the Orbit Scenario File	Computation not performed	XV_CFI_SWATH_POS _ORB_NUM_OEF_ERR	9
ERR	Input time greater than orbital period.	Computation not performed	XV_CFI_SWATH_POS _TIME_ERR	10
ERR	Repeat Days Cycle of this orbit is not the same than the swath template.	Computation not performed	XV_CFI_SWATH_POS _REP_CYCLE_ERR	11

ERR	Orbits Cycle Length of this orbit is not the same than the swath template	Computation not performed	XV_CFI_SWATH_POS _CYCLE_LENGTH_ER R	12
ERR	MLST drift of this orbit is not the same than the swath template.	Computation not performed	XV_CFI_SWATH_POS _MLST_DRIFT_ERR	13
ERR	No spherical triangle.	Computation not performed	XV_CFI_SWATH_POS _SPHER_TRIANG_ER R	14
ERR	Error while transforming from relative to absolute orbit.	Computation not performed	XV_CFI_SWATH_POS _REL_TO_ABS_ERR	15
ERR	Error while computing information of the orbit.	Computation not performed	XV_CFI_SWATH_POS _XV_ORBIT_INFO_ER R	16
ERR	The swath template structure contains invalid data	Computation not performed	XV_CFI_SWATH_POS _SWATH_INIT_ERR	17
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_SWATH_POS _MEMORY_ERR	18
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_SWATH_POS_GE O_SAT_ERR	19
WARN	Deprecated function. Use xv_swathpos_compute instead	Computation performed	XV_CFI_SWATH_POS_DE PRECATED_WARN	20

7.9 xv_swathpos_compute

7.9.1 Overview

The **xv_swathpos_compute** function computes the location of a swath at a given time.

Swath location is expressed as[2]:

- longitude
- latitude
- altitude

for n points (with n >=1). In Figure 2 we can see an example.

xv_swathpos_compute requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data and files, also with precise propagation if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
  - TLE file
- the swath_id (xv_swath_id, initialized using xv_swath_id_init -section 7.32-), providing the Instrument Swath information. If the swath_id is initialized with a Swath definition file or Swath definition data, **xv_swathpos_compute** generates the swath points for the orbit corresponding to input time.

The input time used by **xv_swathpos_compute** is expressed in orbit-relative time.

Users who need to use processing time must make use of the conversion routine provided in EO_VISIBILITY (**xv_time_to_orbit** and **xv_orbit_to_time** functions).

NOTE: Since the swath template file is generated from a reference orbit, it is not allowed to use **xv_swathpos_compute** for an orbit in the orbit scenario file with different repeat cycle or cycle length. If this would happen, **xv_swathpos_compute** will return an error an no computation will be performed.

2 For inertial swaths, right ascension and declination are used instead of longitude and latitude

7.9.2 Calling sequence of xv_swathpos_compute

For C programs, the call to **xv_swathpos_compute** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
      xo_orbit_id orbit_id = {NULL};
      xv_swath_id  swath_id = {NULL};
      xv_time      swathpos_time;
      xv_swath_point_list swath_point_list;
      long ierr[XV_NUM_ERR_SWATHPOS_COMPUTE];

      status = xv_swathpos_compute(&orbit_id,
                           &swath_id, &swathpos_time,
                           &swath_point_list, ierr);
```

7.9.3 Input parameters xv_swathpos_compute

Table 31: Input parameters of xv_swathpos_compute

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
swath_id	xv_swath_id*	-	Swath id	-	-
swathpos_time	xv_time*	-	Input time/orbit for computation.		

7.9.4 Output parameters xv_swathpos_compute

Table 32: Output parameters of xv_swathpos_compute

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_swathpos_compute	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
swath_point_list	xv_swath_point_list*	-	Geodetic information of swath points.		
ierr[XV_NUM_ERR_SWATHPOS_COMPUTE]	long		Error status flags		

7.9.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_swathpos_compute** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_swathpos_compute** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 33: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong orbit Id.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_ORBIT_STATUS_ERR	0
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_ORBIT_MODEL_ERR	1
ERR	Orbital information does not coincide with reference swath.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_INCONSISTENT_SWATH_ERR	2
ERR	Orbit number must be positive.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_ORB_NUM_LIM_ERR	3
ERR	Seconds since ascending node must be zero or positive.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_SEC_LIM_ERR	4
ERR	MicroSeconds must be zero or positive.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_MICROSEC_1ST_ERR	5
ERR	MicroSeconds can not be bigger than 999999.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_MICROSEC_2ND_ERR	6
ERR	Orbit type switch out of range.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_ORBIT_TYPE_ERR	7
ERR	Cycle number must be positive.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_CYCLE_ERR	8
ERR	Orbit number is not included in the Orbit Scenario File.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_ORB_NUM_OEF_ERR	9

ERR	Input time greater than orbital period.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_TIME_ERR	10
ERR	Repeat Days Cycle of this orbit is not the same than the swath template.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_REP_CYCLE_ERR	11
ERR	Orbits Cycle Length of this orbit is not the same than the swath template.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_CYCLE_LENGTH_ERR	12
ERR	MLST drift of this orbit is not the same than the swath template.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_MLST_DRIFT_ERR	13
ERR	No spherical triangle.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_SPHER_TRIANG_ERR	14
ERR	Error while transforming from relative to absolute orbit.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_REL_TO_ABS_ERR	15
ERR	Error while computing information of the orbit.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_XV_ORBIT_INFO_ERR	16
ERR	The swath template structure contains invalid data	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_SWATH_INIT_ERR	17
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_MEMORY_ERR	18
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_GEO_SAT_ERR	19
ERR	Wrong input time type. Orbit data must be provided.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_TIME_TYPE_ERR	20
ERR	Error generating Swath Template File	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_GEN_SWATH_ERR	21
ERR	Error reading Swath File: %s	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_READ_SWATH_FILE_ERR	22
WARN	Orbit outside STF validity but no recomputation can be performed.	Computation performed	XV_CFI_SWATHPOS_COMPUTE_SWATH_VALIDITY_WARN	23
ERR	Error transforming time to orbit.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_TIME_TO_ORBIT_ERR	24
ERR	Input file is not a swath file.	Computation not performed	XV_CFI_SWATHPOS_COMPUTE_DETECT_SWATH_TYPE_ERR	25

7.10 xv_star_vis_time

7.10.1 Overview

The **xv_star_vis_time** function computes stars visibility segments, the orbital segments for which a given star is visible with a given instrument from the satellite.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as seconds elapsed since the ascending node crossing.

In addition, **xv_star_vis_time** calculates for every start and end of the visibility segment a coverage flag, determining which side of the FOV the event took place.

xv_star_vis_time requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files, also with precise propagation if applicable (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
  - TLE files

  Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.
- Two Inertial Reference Swath Files. The Swath data can be provided by:
  - A swath template file produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).
  - A swath definition file, describing the swath geometry. In this case the **xv_star_vis_time** generates the swath points for a number of orbits given by the user.
- (*Optional*) The Star's Database File, describing the location in right ascension and declination of a star, described by its corresponding identifier.

The time intervals used by **xv_star_vis_time** are expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter "Orbit Range": first and last orbit to be considered. In case of using relative orbits, the corresponding cycle number should be used, otherwise, this the cycle number will be a dummy parameter
- output parameter "Star Visibility Segments": time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microsecs}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

Users who need to use processing times must make use of the conversion routines provided in EO_VISIBILITY (**xv_time_to_orbit** and **xv_orbit_to_time** functions).

7.10.2 Swath Definition

xv_star_vis_time calculates stars visibility segments for FOV corresponding to limb-sounding instruments observing inertial objects. The corresponding template files are generated off-line by the EO_VISIBILITY CFI software (**xv_gen_swath** function).

7.10.2.1 Inertial Swaths

The FOV for a Limb-sounding instrument observing inertial objects is calculated using two main parameters.

- The FOV projection on the celestial sphere is determined by two set of swaths, one corresponding to a higher (TOP) and a lower (BOTTOM) altitude over the ellipsoid, hence defining the elevation range of the FOV
- The azimuth range is defined as such, the extremes corresponding to the left and right sides. In addition **xv_gen_swath** generates coordinates for a middle point

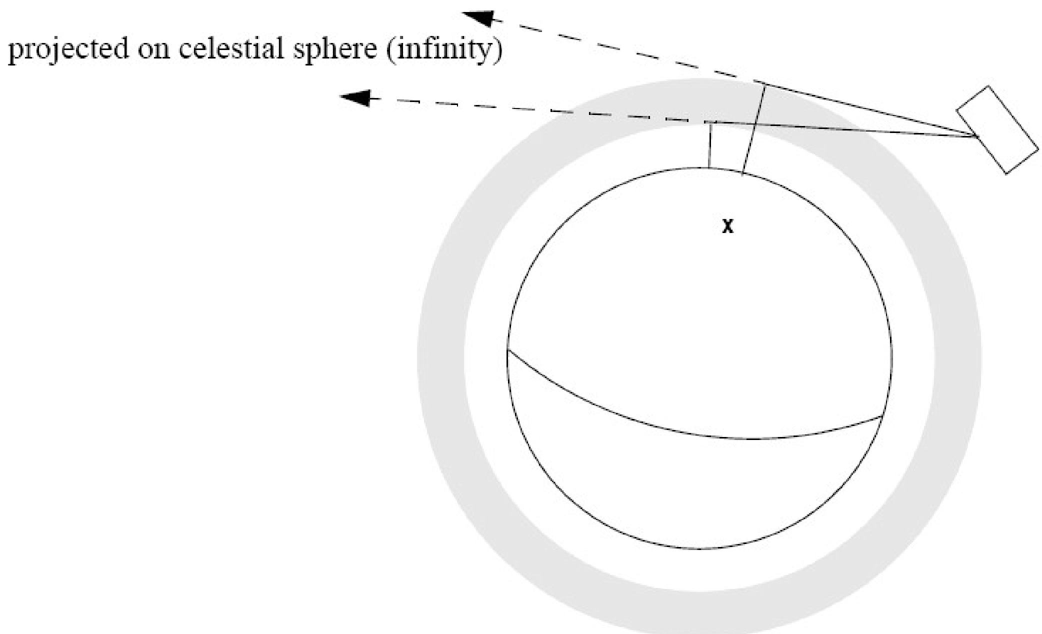


Figure 19: Two tangent altitudes over the ellipsoid

The instantaneous FOV projected on the celestial sphere can be represented as a series of points defined by their Right Ascension and Declination coordinates.

The top and bottom lines sweep the azimuth range at a constant tangent altitude, whilst the left and right side have a constant azimuth value with changing tangent altitude.

The shape of FOV should be similar to that shown in the diagram below with the dotted lines, whilst the algorithm implemented in xv_star_vis_time uses a simplified model joining the points with straight line.

As the satellite evolves around the orbit and the FOV sweeps the celestial sphere, a star can enter the FOV. **xv_star_vis_time** calculates that time and returns a flag indicating which part of the FOV (*LEFT*, *TOP*, *RIGHT* or *BOTTOM*) first detected the star. The same is done when the star exits the FOV.

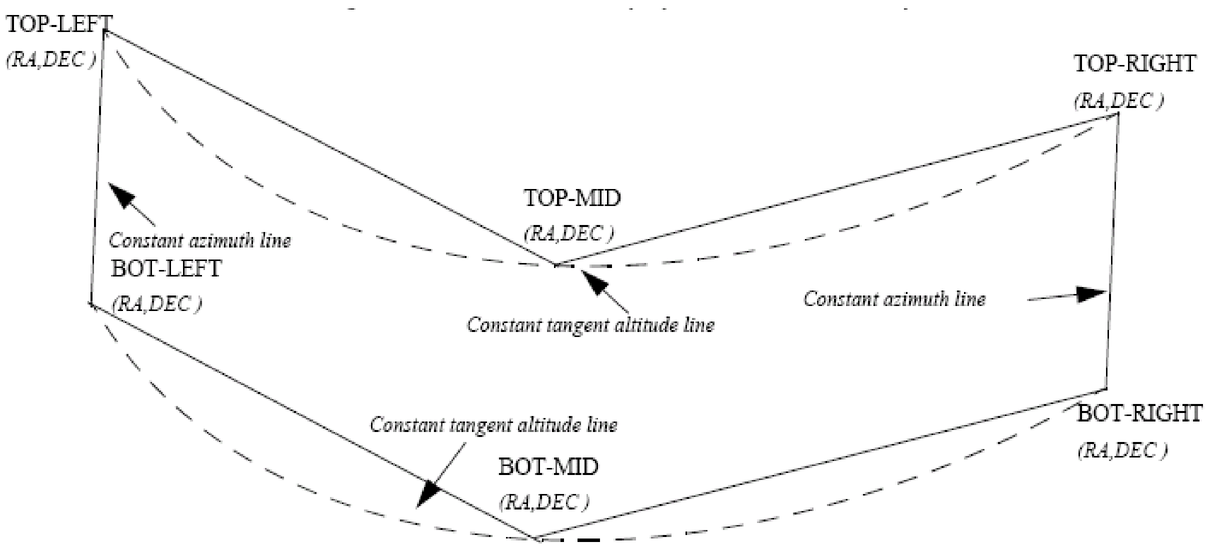


Figure 20: Instantaneous FOV projected on the celestial sphere

7.10.2.2 Splitting swaths

As it was shown in Figure 20, the accuracy and precision of **xv_star_vis_time** strongly depends on how close the projection used in the algorithm is to the real world. Higher accuracy can be obtained splitting the azimuth range in sub-swaths.

Furthermore, splitting the swath would be necessary if the FOV was to cover an azimuth range larger than 180 degrees.

Note: It is important to note that if the FOV covers the value of 90 or 270 degrees in azimuth, one of the extremes (*LEFT* or *RIGHT*) of the STF must correspond to that azimuth value.

7.10.2.3 Orbital Changes

If **xv_star_vis_time** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behaviour depends on the swath files introduced as input:

•If **swath template files** are used, **xv_star_vis_time** automatically will ignore the orbits that do not correspond with the template files (i.e. no visibility segments will be generated for those orbits), since swath template files are generated from a reference orbit with a particular geometry, so they are not valid for a different geometry.

•If **swath definition files** are introduced, **xv_station_vis_time** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath templates valid for every orbital change are generated to perform the calculations.

7.10.2.4 Format of Swath Template File

If a swath template file with the variable header tags *Start_Validity_Range* and *Stop_Validity_Range* is used as input, only the segments belonging to that orbit range will be returned.

7.10.2.5 MLST non linear drift

If a swath definition file is introduced, it can be also introduced every how many orbits the swath template file must be recomputed (swath_flag parameter, see section 164). If the orbit_id has been initialized with an OSF file with MLST non linear terms and the parameter swath_flag is greater than the linear approximation validity, the recomputation of swath template file will be done every linear approximation validity orbits.

7.10.3 Calling sequence xv_star_vis_time

For C programs, the call to **xv_star_vis_time** is (<u>input</u> parameters are <u>underlined</u>):

```
#include"explorer_visibility.h"
{
      xo_orbit_id  orbit_id = {NULL};
      long         swath_flag, orbit_type,
                   start_orbit, start_cycle,
                   stop_orbit, stop_cycle,
                   number_segments,
                   *bgn_orbit, *bgn_second, *bgn_microsec,
                   *bgn_cycle, *bgn_coverage,
                   *end_orbit, *end_second, *end_microsec,
                   *end_cycle, *end_coverage,
                   ierr[XV_NUM_ERR_STAR_VIS_TIME], status;
      double           star_ra, star_dec, star_ra_deg, star_dec_deg,
                   min_duration;
      char         *orbit_scenario_file,
                   *swath_file_upper, *swath_file_lower;
      char         star_id[8],  *star_db_file;

      status = xv_star_vis_time(
                       &orbit_id, &orbit_type,
                       &start_orbit, &start_cycle,
                       &stop_orbit, &stop_cycle,
                       &swath_flag, swath_file_upper, swath_file_lower,
                       star_id, star_db_file,
                       &star_ra, &star_dec,
                       &min_duration,
                       &star_ra_deg, &star_dec_deg,
                       &number_segments,
                       &bgn_orbit, &bgn_second, &bgn_microsec,
                       &bgn_cycle, &bgn_coverage,
                       &end_orbit, &end_second, &end_microsec,
                       &end_cycle, &end_coverage,
                       ierr);
```

```
        /* Or, using the run_id */
        long run_id;

        status = xv_star_vis_time_run(
                          &run_id, &orbit_type,
                          &start_orbit, &start_cycle,
                          &stop_orbit, &stop_cycle,
                          &swath_flag, swath_file_upper, swath_file_lower,
                          star_id, star_db_file,
                          &star_ra, &star_dec,
                          &min_duration,
                          &star_ra_deg, &star_dec_deg,
                          &number_segments,
                          &bgn_orbit, &bgn_second, &bgn_microsec,
                          &bgn_cycle, &bgn_coverage,
                          &end_orbit, &end_second, &end_microsec,
                          &end_cycle, &end_coverage,
                          ierr);
}
```

7.10.4 Input parameters xv_star_vis_time

Table 34: Input parameters of xv_star_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) If set to zero then first orbit of orbit_scenario_file is selected.	absolute or relative orbit number	= 0 or: absolute orbits ≥start_osf relative orbits ≤repeat cycle
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ start_osf
stop_orbit	long	-	Last orbit, segment filter. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle) If it is not initialized with orbital changes, stop orbit will be set to the last orbit in orbit_id initialization.	absolute or relative orbit number	= 0 or: absolute orbits ≥start_osf relative orbits ≤ repeat cycle
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	=0 or ≥ start_osf
swath_flag	long*	-	Define the use of the swath file: • 0 = (XV_STF) if the swath file is a	-	XV_STF = 0 XV_SDF = 1

			swath template file. • > 0 if the swath files is a swath definition file. In this case the swath points are generated for every “swath_flag” orbits		> 0
swath_file_uppe r	char *		File name of the inertial swath-file for the appropriate instrument mode, which defines the upper limit of the FOV. This file is read each time the function is called		
swath_file_lowe r	char *		File name of the inertial swath-file for the appropriate instrument mode, which defines the lower limit of the FOV. This file is read each time the function is called		
star_id[8]	char		identification of the star, as defined in the star_db_file. This parameter is used **ONLY IF** star_db_file is not equal empty string(““)		EXACTLY 8 characters
star_db_file	char *		File name of the star database file		
star_ra	double*		Right Ascension of Star, in TOD. This parameter is used **ONLY IF** star_db_file is equal empty string (““)	deg	(-180.0, 180.0)
star_dec	double*		Declination of Star, in TOD. This parameter is used **ONLY IF** star_db_file is equal empty string (““)	deg	(-90.0, 90.0)
min_duration	double*		Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0.0

7.10.5 Output parameters xv_star_vis_time

Table 35: Output Parameters of xv_star_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_star_vis_time	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
star_ra_deg	double		Right Ascension of the star, in TOD, for the UTC halfway start_orbit and stop_orbit.	deg	(-180.0, 180.0)
star_dec_deg	double		Declination of the star, in TOD, for the UTC halfway start_orbit and stop_orbit.	deg	(-90.0, 90.0)
number_segment	long		Number of visibility segments returned to the user		≥ 0
bgn_orbit	long*	all	Orbit number, begin of visibility segment i bgn_orbit[i-1], i = 1, number_segments		> 0
bgn_second	long*	all	Seconds since ascending node, begin of visibility segment i bgn_second[i-1], i = 1, number_segments	s	≥ 0 < orbital period
bgn_microsec	long*	all	Micro seconds within second begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments	µs	≥ 0 ≤ 999999
bgn_cycle	long*	all	cycle number begin of visibility segment i bgn_microsec[i-1], i = 1, number_segments		> 0 NULL when using relative orbits
bgn_coverage	long*	all	Coverage flag for swath entry: XV_STAR_UNDEFINED = 0, XV_STAR_UPPER = 1, XV_STAR_LOWER = 2, XV_START_LEFT = 3,		0,1,2,3,4

			XV_STAR_RIGHT=4		
end_orbit	long*	all	Orbit number, end of visibility segment i end_orbit[i-1], i = 1, number_segments		> 0
end_second	long*	all	Seconds since ascending node, end of visibility segment i end_second[i-1], i = 1, number_segments	s	≥ 0 <orbital period
end_microsec	long*	all	Micro seconds within second end of visibility segment i end_microsec[i-1], i = 1, number_segments	µs	0 ≤ 999999
end_cycle	long*	all	End cycle, end of visibility segment i end_orbit[i-1], i = 1, number_segments		>0 NULL when using relative orbits
end_coverage	long*	all	Coverage flag for swath exit: XV_STAR_UNDEFINED = 0, XV_STAR_UPPER = 1, XV_STAR_LOWER = 2, XV_START_LEFT = 3, XV_STAR_RIGHT=4		0,1,2,3,4
ierr[10]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_star_vis_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.10.6 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_star_vis_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_star_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 36: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error, wrong orbit Id.	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_STATUS_ERR	0
WAR N	Warning, start orbit is outside range of OEF/OSF.	Computation performed Message to inform the user	XV_CFI_STAR_VIS_TIME_FIRST_ORBIT_WARN	1
WAR N	Warning, stop orbit is outside range of OEF/OSF.	Computation performed Message to inform the user	XV_CFI_STAR_VIS_TIME_LAST_ORBIT_WARN	2
WAR N	Warning, there is an orbital change within the requested orbits.	Computation performed Message to inform the user	XV_CFI_STAR_VIS_TIME_ORBIT_CHANGE_WA RN	3
ERR	Error, starvistime can only operate with an inertial swath.	Computation not performed	XV_CFI_STAR_VIS_TIME_INERTIAL_SWATH_E RR	4
ERR	Error, Orbital information does not coincide with reference swath.	Computation not performed	XV_CFI_STAR_VIS_TIME_INCONSISTENT_SWATH_ERR	5
ERR	Input parameter \"swath_flag\" is out of range	Computation not performed	XV_CFI_STAR_VIS_TIME_SWATH_FLAG_ERR	6
ERR	Could not generate the swath template data	Computation not performed	XV_CFI_STAR_VIS_TIME_GENSWATH_ERR	7
ERR	Low swath altitude is above the upper limit described by the higher swath altitude.	Computation not performed	XV_CFI_STAR_VIS_TIME_ALT_ERR	8
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_STAR_VIS_TIME_INTERNAL_MEMORY_ERR	9
ERR	Error allocating memory for the	Computation not	XV_CFI_STAR_VIS_TIM	10

	visibility segments.	performed	E_SEGMENTS_MEMORY_ERR	
ERR	Error allocating memory for the coverage.	Computation not performed	XV_CFI_STAR_VIS_TIME_COVERAGE_MEMORY_ERR	11
ERR	Input parameter "orbit_type" is out of range.	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_TYPE_ERR	12
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_MODEL_ERR	13
ERR	Error in input parameter to starvistime.	Computation not performed	XV_CFI_STAR_VIS_TIME_INPUTS_CHECK_ERR	14
ERR	Error while transforming into absolute orbit the start_orbit.	Computation not performed	XV_CFI_STAR_VIS_TIME_REL_TO_ABS_START_ERR	15
ERR	Error while transforming into absolute orbit the stop_orbit.	Computation not performed	XV_CFI_STAR_VIS_TIME_REL_TO_ABS_STOP_ERR	16
ERR	Error updating star's position in from JD2000 to determined UTC.	Computation not performed	XV_CFI_STAR_VIS_TIME_STAR_RADEC_ERR	17
ERR	Error obtaining orbital information.	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_INFO_ERR	18
ERR	Error reading the upper swath template file.	Computation not performed	XV_CFI_STAR_VIS_TIME_SWATH_UPPER_READ_ERR	19
ERR	Error reading the lower swath template file.	Computation not performed	XV_CFI_STAR_VIS_TIME_SWATH_LOWER_READ_ERR	20
ERR	Error reading the star data file.	Computation not performed	XV_CFI_STAR_VIS_TIME_READ_STAR_ERR	21
ERR	Error determining transitions.	Computation not performed	XV_CFI_STAR_VIS_TIME_STAR_MAIN_ERR	22
ERR	Error while transforming into relative orbits the output segments.	Computation not performed	XV_CFI_STAR_VIS_TIME_ABS_TO_REL_ERR	23
ERR	Error transforming orbit to time.	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_TO_TIME_ER R	24
ERR	Error reading the swath definition file: %s	Computation not performed	XV_CFI_STAR_VIS_TIME_READ_SDF_ERR	25
ERR	Error checking orbital change	Computation not	XV_CFI_STAR_VIS_CHECK_	26

	in range	performed	ORBITAL_CHANGE_ERR	
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_INTERVAL_STF_ERR	27
WARN	Input orbit interval is partially outside STF validity interval	Computation performed	XV_CFI_STAR_VIS_TIME_ORBIT_INTERVAL_STF_WARN	32
WARN	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation performed	XV_CFI_STAR_VIS_TIME_OSF_NON_LIN_STF_OLD_WARN	33
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used	Computation performed	XV_CFI_STAR_VIS_TIME_SWATH_FLAG_LARGER_THAN_LIN_APPROX_VAL_WARN	34
WARN	Visibility computations with precise propagator can be very slow	Computation performed	XV_CFI_STAR_VIS_TIME_PRECISE_PROPAG_WARN	35

7.11xv_multizones_vis_time

7.11.1 Overview

Note: this function is deprecated. Use **xv_zonevistime_compute** instead.

The **xv_multizones_vis_time** function computes all the orbital segments for which a given instrument swath intercepts several user-defined zones at the surface of the Earth ellipsoid.

The visibility segments are obtained by calling to **xv_zone_vis_time** (see section 7.1 for further details about swaths, zones and visibility segments definitions). Those segments are merged and ordered by start time. In addition to this, two tables are provided. The first one contains the zones where segment has visibility, and the second one contains the coverage of the segment for each zone (see Figure 21).

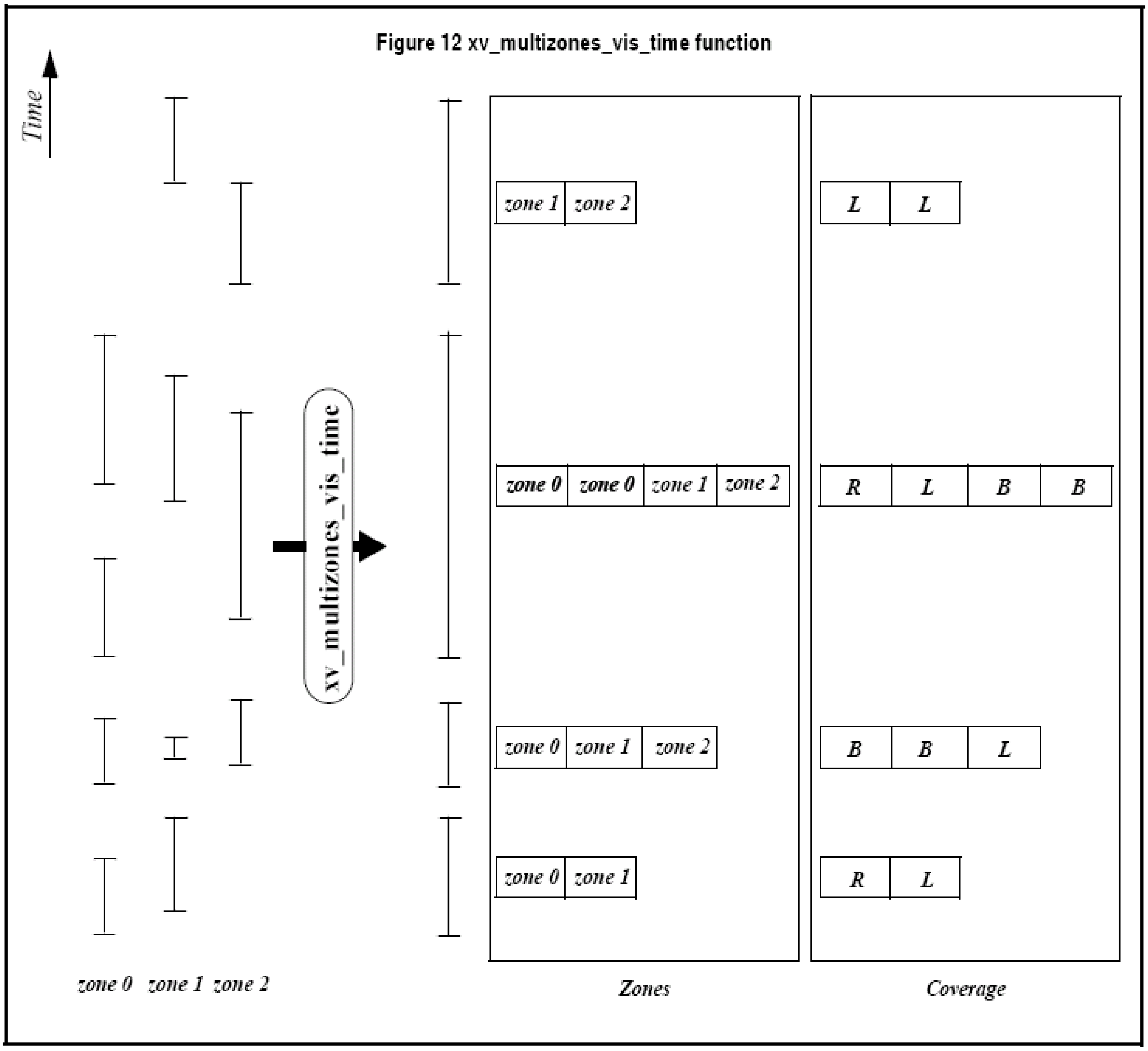


Figure 21: xv_multizones_vis_time function

The time intervals used by **xv_multizones_vis_time** are expressed in absolute orbit numbers or in relative orbit and cycle numbers. This is valid for both:

- input parameter "Orbit Range": first and last orbit to be considered. In case of using relative orbits, the corresponding cycle number should be used, otherwise, this the cycle number will be a dummy parameter.
- output parameter "Zone Visibility Segments": time segments with time expressed as {absolute orbit number (or relative orbit number and cycle number), number of seconds since ascending node, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits.

xv_multizones_vis_time requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files

  Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.

- the Instrument Swath File, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit. The Swath data can be provided by:
  - A swath template file produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).
  - A swath definition file, describing the swath geometry. In this case the **xv_multizones_vis_time** generates the swath points for a number of orbits given by the user.

• optionally, a Zone Database File, containing the zone description. The user can either specify a zone identifier referring to a zone in the file, or provide the zone parameters directly to **xv_multizones_vis_time.**

Users who need to use processing times must make use of the conversion routines provided in EO_ORBIT (**xo_time_to_orbit** and **xo_orbit_to_time** functions).

NOTE: If **xv_multizones_vis_time** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behaviour depends on the swath files introduced as input:

•If a **swath template file** is used, **xv_multizones_vis_time** automatically will ignore the orbits that do not correspond with the template file (i.e. no visibility segments will be generated for those orbits), since swath template file is generated from a reference orbit with a particular geometry, so it is not valid for a different geometry.

•If a **swath definition file** is introduced, **xv_multizones_vis_time** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath templates valid for every orbital change are generated to perform the calculations.

7.11.2 Calling sequence xv_multizones_vis_time

For C programs, the call to **xv_multizones_vis_time** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

      xo_orbit_id orbit_id = {NULL};
      long     swath_flag, orbit_type,
               start_orbit, start_cycle, stop_orbit, stop_cycle,
               num_zones, projection, *zone_num,
               extra_info_flag,
               number_segments,
               *bgn_orbit, *bgn_secs, *bgn_microsecs, *bgn_cycle,
               *end_orbit, *end_secs, *end_microsecs, *end_cycle,
               *nb_zon_in_segment, **zones_in_segment, **coverage,
               ierr[XV_NUM_ERR_MULTIZONES_VIS_TIME], status;

      double   *zone_long, *zone_lat, *zone_diam,
               min_duration;

      char     *swath_file, *zone_db_file,
               **zone_id;

      status = xv_multizones_vis_time(
                    &orbit_id, &orbit_type,
                    &start_orbit, &start_cycle,
                    &stop_orbit, &stop_cycle,
                    &swath_flag, swath_file, &num_zones,
                    zone_id, zone_db_file,
                    projection, zone_num,
                    zone_long, zone_lat, zone_diam,
                    &min_duration, &extra_info_flag,
                    &number_segments,
                    &bgn_orbit, &bgn_second, &bgn_microsec, &bgn_cycle,
                    &end_orbit, &end_second, &end_microsec, &end_cycle,
                    &nb_zon_in_segment, &zones_in_segment, &coverage,
                    ierr);
```

```
        /* Or, using the run_id */
        long run_id;

        status = xv_multizones_vis_time_run(
                     &run_id, &orbit_type,
                     &start_orbit, &start_cycle,
                     &stop_orbit, &stop_cycle,
                     &swath_flag, swath_file, &num_zones,
                     zone_id, zone_db_file,
                     projection, zone_num,
                     zone_long, zone_lat, zone_diam,
                     &min_duration, &extra_info_flag,
                     &number_segments,
                     &bgn_orbit, &bgn_second, &bgn_microsec, &bgn_cycle,
                     &end_orbit, &end_second, &end_microsec, &end_cycle,
                     &nb_zon_in_segment, &zones_in_segment, &coverage,
                     ierr);
}
```

7.11.3 Input parameters xv_multizones_vis_time

Table 37: Input parameters of xv_multizones_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤ repeat cycle
• start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤ repeat cycle
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
swath_flag	long*	-	Define the use of the swath file: • 0 = (XV_STF) if the swath file is a swath template file. • > 0 if the swath files is a swath	-	XV_STF = 0 XV_SDF = 1 > 0

			definition file. In this case the swath points are generated for every “swath_flag” orbits		
swath_file	char *	-	File name of the swath-file for the appropriate instrument mode		
num_zones	long	-	Number of zones		>0
zone_id	char**	all	Identification name for n-th zone (0<n<num_zones). It must exist for every zone. zone_id[i] must belong to a zone from the zone_db_file when zone_num[i]=0.		
zone_db_file	char *	-	File name of the zone-database file. Dummy when no zones from database are selected.		
projection	long*	all	projection for each zone used to define polygon sides as straight lines.		complete. See Table 3 (Projections)
zone_num	long*	all	Number of vertices of the n-th zone (0<n<num_zones) provided in zone_long, zone_lat: = 0 no vertices provided, use zone_id / zone_db_file = 1 Point / Circular zone, = 2 Line zone > 2 Polygon zone		≥ 0
zone_long	double*	all	Geocentric longitude of - circle centre, for circ. zone - point, for point zone - line-end, for line zone - vertices, for polygon zone. The longitude of the vertices corresponding to all zones shall be arranged consecutively[3].	deg	
zone_lat	double*	all	Geodetic latitude of - circle centre, for circ. zone.	deg	

3 For example,
- zone 0: points will be arranged from 0 to zone_num[0] (no points in case of using a database zone),
- zone 1: points will be arranged from zone_num[0] to zone_num[0] + zone_num[1]
- ...

			- point, for point zone. - line-end, for line zone. - vertices, for polygon zone. The latitude of the vertices corresponding to all zones shall be arranged consecutivelySuperscriptparanum.		
zone_diam	double*	all	Array of diameters of circular zones in case this shape is selected for any zone[4]. zone_diam=0.0 for Point Zones.	m	≥ 0.0
min_duration	double	-	Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0
extra_info_fla g	long	-	If value set to false (= 0), the zones_in_segment and coverage arrays are not computed. Saves computation time.		0 (false), 1 (true)

4 The values corresponding to all zones shall be arranged consecutively, so that the zone_diam[0] corresponds with the first point or circular zone, zone_diam[1] corresponds with the second point or circular zone, and so on.

7.11.4 Output parameters xv_multizones_vis_time

Table 38: Output parameters of xv_multizones_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_multizones_vis_ti me	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long	-	Number of segments in the output lists.	-	> 0
bgn_orbit	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_second	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsec	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_second	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsec	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
nb_zon_in_segment	long*	all	Number of zones where the segment has visibility. Dummy if extra_info_flag=0 (false).	-	>0
zones_in_segment	long**	all	Index of the zone_id input array where the segment has visibility. Dummy if extra_info_flag=0 (false).	-	≥0
coverage	long**	all	Coverage of the segment in each of the zones. Dummy if extra_info_flag=0 (false).		complete See Table 3
ierr	long*		Error status flags		

Note 1: The zones_in_segment and coverage arrays are returned as a two-dimensional table where the first index is related to the output visibility segment , and the second one goes all over the zones that compose that segment.

Note2 (Memory Management): Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_multizones_vis_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.11.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_multizones_vis_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_multizones_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 39: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_MEMORY_ERR	0
ERR	Error getting visibility segments for zone %ld	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_COMPUTE_SEGMENTS_ERR	1
ERR	Error getting absolute orbit from relative orbit	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_ABS_TO_REL_ORBIT_ERR	2
ERR	Error getting relative orbit vector from absolute orbits	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_ABS_TO_REL_VECTOR_ERR	3
ERR	Error while merging overlapped segments	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_OVERLAP_ERR	4
ERR	Orbit id is not initialized.	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_ORBIT_STATUS_ERR	5
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_MULTIZONES_VIS_TIME_GEO_SAT_ERR	6
WARN	Deprecated function. Use xv_zonevistime_compute instead	Computation performed	XV_CFI_MULTIZONES_DEPRECATED_WARN	7

7.12 xv_multistations_vis_time

7.12.1 Overview

Note: this function is deprecated. Use **xv_stationvistime_compute** instead.

The **xv_multistations_vis_time** function computes visibility segments of several ground stations, i.e. the orbital segments for which the satellite is visible from a ground station located at the surface of the Earth.

The visibility segments are obtained by calling to **xv_station_vis_time**. Those segments are merged and ordered by start time. Moreover, **xv_multistations_vis_time** provides a table containing the stations from which the satellite is visible in each segment.

In addition, **xv_multistations_vis_time** computes the time of zero-doppler (i.e. the time at which the range-rate to the station is zero) per station.

The time intervals used by **xv_multistations_vis_time** are expressed in absolute orbit numbers or in relative orbit and cycle numbers. This is valid for both:

- input parameter "Orbit Range": first and last orbit to be considered. In case of using relative orbits, the corresponding cycle number should be used, otherwise, this the cycle number will be a dummy parameter.
- output parameter "Stations Visibility Segments": time segments with time expressed as {absolute orbit number (or relative orbit number and cycle number), number of seconds since ascending node, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits.

xv_multistations_vis_time requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files

  Note: if the orbit is initialized for precise propagation, the execution of the visibility function can be very slow. As alternative, a POF can be generated with the precise propagator (function xo_gen_pof) for the range of orbits the user usually needs, and use this generated file to initialize the orbit id. The execution time performance will be much better for the visibility function and it will not have a big impact on the precision of the calculations.

- the Instrument Swath File, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit. The Swath data can be provided by:

- A swath template file produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).

- A swath definition file, describing the swath geometry. In this case the **xv_multistations_vis_time** generates the swath points for a number of orbits given by the user.

- the Instrument Swath File, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit. It is produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).
- the Station Database File, describing the location and the physical mask of each ground station.

Users who need to use processing times must make use of the conversion routines provided in EO_ORBIT (**xo_time_to_orbit** and **xo_orbit_to_time** functions).

NOTE: If **xv_multistation_vis_time** is used with a range of orbits that includes an orbital change (e.g. change in the repeat cycle or cycle length), the behaviour depends on the swath files introduced as input:

•If a **swath template file** is used, **xv_multistation_vis_time** automatically will ignore the orbits that do not correspond with the template file (i.e. no visibility segments will be generated for those orbits), since swath template file is generated from a reference orbit with a particular geometry, so it is not valid for a different geometry.

•If a **swath definition file** is introduced, **xv_multistation_vis_time** will perform the computations across orbital changes, and will return the visibility segments corresponding to the whole orbital range. Internally, swath templates valid for every orbital change are generated to perform the calculations.

7.12.2 Calling sequence xv_multistations_vis_time

For C programs, the call to **xv_multistations_vis_time** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
        xo_orbit_id orbit_id = {NULL};
        long     swath_flag, orbit_type,
                 start_orbit, start_cycle,
                 stop_orbit, stop_cycle,
                 num_stations, *mask,
                 extra_info_flag,
                 number_segments,
                 *bgn_orbit, *bgn_secs, *bgn_microsecs, *bgn_cycle,
                 *end_orbit, *end_secs, *end_microsecs, *end_cycle,
                 **zdop_orbit, **zdop_secs, **zdop_microsecs, **zdop_cycle,
                 *nb_stat_in_segment, **stat_in_segment,
                 ierr[XV_NUM_ERR_MULTISTATIONS_VIS_TIME], status;

        double   *aos_elevation, *los_elevation,
                 min_duration;

        char     *swath_file, *station_db_file,
                 **station_id;

        status = xv_multistations_vis_time(
                      &orbit_id, &orbit_type,
                      &start_orbit, &start_cycle,
                      &stop_orbit, &stop_cycle,
                      &swath_flag, swath_file, &num_stations,
                      station_db_file, station_id,
                      aos_elevation, los_elevation, mask,
                      &min_duration,
                      &extra_info_flag,
                      &number_segments,
                      &bgn_orbit, &bgn_second, &bgn_microsec, &bgn_cycle,
                      &end_orbit, &end_second, &end_microsec, &end_cycle,
                      &zdop_orbit, &zdop_second,&zdop_microsec,&zdop_cycle,
                      &nb_stat_in_segment, &stat_in_segment,
                      ierr);
```

```
        /* Or, using the run_id */
        long run_id;

        status = xv_multistations_vis_time_run(
                        &run_id, &orbit_type,
                        &start_orbit, &start_cycle,
                        &stop_orbit, &stop_cycle,
                        &swath_flag, swath_file, &num_stations,
                        station_db_file, station_id,
                        aos_elevation, los_elevation, mask,
                        &min_duration,
                        &extra_info_flag,
                        &number_segments,
                        &bgn_orbit, &bgn_second, &bgn_microsec, &bgn_cycle,
                        &end_orbit, &end_second, &end_microsec, &end_cycle,
                        &zdop_orbit, &zdop_second,&zdop_microsec,&zdop_cycle,
                        &nb_stat_in_segment, &stat_in_segment,
                        ierr);
}
```

7.12.3 Input parameters xv_multistations_vis_time

Table 40: Input parameters of xv_multistations_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
start_orbit	long	-	First orbit, segment filter Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤ repeat cycle
• start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤ repeat cycle
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
swath_flag	long*	-	Define the use of the swath file: • 0 = (XV_STF) if the swath file is a swath template file. • > 0 if the swath files is a swath	-	XV_STF = 0 XV_SDF = 1 > 0

			definition file. In this case the swath points are generated for every "swath_flag" orbits		
swath_file	char *	-	File name of the swath-file for the appropriate instrument mode		
num_stations	long	-	Number of stations		>0
station_db_file	char *	-	File name of the station-database file.		
station_id	char**	-	Identification name for n-th station (0<n<num_stations).		
aos_elevation	double*	all	Minimum elevation to consider at AOS for each station(i.e. before considering start of visibility).	deg	≥ 0.0
los_elevation	double*	all	Maximum elevation to consider at LOS for each station(i.e. before considering end of visibility).	deg	≥ 0.0 ≤ aos_elevation
mask	long*	all	mask used to define visibility = 0 combine AOS/LOS elevations and physical mask (nominal mode) = 1 consider only AOS/LOS elevations = 2 consider only physical mask		≥ 0
min_duration	double	-	Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥ 0
extra_info_flag	long	-	If value set to false (= 0), the zero doppler arrays and stations arrays are not computed. Saves computation time.		0(false), 1 (true)

7.12.4 Output parameters xv_multistations_vis_time

Table 41: Output parameters of xv_multistations_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_multistations_vis_time	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
number_segments	long	-	Number of segments in the output lists.	-	> 0
bgn_orbit	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_second	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsec	long*	all	Array of micro seconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_second	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsec	long*	all	Array of micro seconds within a second for the end of the segments	-	>0 <999999
end_cycle	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
zdop_orbit	long**	all	Orbit number, time of zero doppler for each segment and zone (-1 if no zero doppler within corresponding visibility segment) Dummy if extra_info_flag = false.		> 0
zdop_second	long**	all	Seconds since ascending node, time of zero doppler for each segment and zone (-1 if no zero doppler within corresponding visibility segment) Dummy if extra_info_flag = false.	s	>= 0 < orbital period

zdop_microsec	long**	all	Micro seconds within second time of zero doppler for each segment and zone (-1 if no zero doppler within corresponding visibility segment) Dummy if extra_info_flag = false.	µs	0 =< =< 999999
zdop_cycle	long**	all	Cycle number, time of zero doppler for each segment and zone (-1 if no zero doppler within corresponding visibility segment) Dummy if extra_info_flag = false.		>0 NULL when using absolute orbits
nb_stat_in_segment	long*	all	nb_stat_in_segment [i] =Number of stations from which the satellite is visible during the i-th segment of time. Dummy if extra_info_flag = false.	-	>0
stat_in_segment	long**	all	stat_in_segment [i] = array of indexes of the stations from which the satellite is visible during the i-th segment. Dummy if extra_info_flag = false.	-	≥0
ierr	long*		Error status flags		

Note 1: The stat_in_segment and zdop_xxx arrays are returned as a two-dimensional table where the first index is related to the output visibility segment , and the second one goes all over the zones that compose that segment.

Note 2 (Memory Management): Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_multistations_vis_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.12.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_multistations_vis_time** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_multistations_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 42: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_MEMORY_ERR	0
ERR	Error getting visibility segments for station %ld	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_COMPUTE_SEGMENTS_ER R	1
ERR	Error getting absolute orbit from relative orbit	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_ABS_TO_REL_ORBIT_ERR	2
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_ABS_TO_REL_VECTOR_ER R	3
ERR	Error while merging overlapped segments.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_OVERLAP_ERR	4
ERR	Orbit id not initialized.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_ORBIT_STATUS_ERR	5
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_GEO_SAT_ERR	6
WARN	Deprecated function. Use xv_stationvistime_compute instead	Computation performed	XV_CFI_MULTISTATIONS_VIS_TIME_DEPRECATED_WARN	7

7.13 xv_orbit_extra

7.13.1 Overview

The **xv_orbit_extra** function computes for an input orbit, the times for:

- an input set of Sun zenit angles are reached (both up and down times are computed)
- Sun ocultations by the Earth.
- Sun ocultations by the Moon.

xv_orbit_extra needs as input the orbital parameters returned by **xo_orbit_info** (its output array result_vector). So, the natural use to call to **xv_orbit_extra** will be:

- Initialise time references: calling to **xl_time_ref_init** of **xl_time_ref_init_file**.
- Orbital initialisation by calling one of the functions: **xo_orbit_init_file**, **xo_orbit_init_def** or **xo_orbit_cart_init**.
- Call to **xo_orbit_info** to get the result_vector containing the orbital parameters of the orbit.
- Call to **xv_orbit_extra** with the same orbit than in the call to the orbit_info function.

The input orbit must be an absolute orbit.

Users who need to use processing times must make use of the conversion routines provided in EO_ORBIT (**xo_time_to_orbit** and **xo_orbit_to_time** functions).

7.13.2 Calling sequence xv_orbit_extra

For C programs, the call to **xv_orbit_extra** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

    xo_orbit_id orbit_id = {NULL};
    long      orbit,
              num_sza,
              ierr[XV_NUM_ERR_ORBIT_EXTRA];
    double    orbit_info_vector[XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS], *sza,
              *sza_up, *sza_down,
              eclipse_entry, eclipse_exit,
              sun_moon_entry, sun_moon_exit;

    status= xv_orbit_extra (&orbit_id, &orbit, orbit_info_vector,
                      &num_sza, sza,
                      &sza_up, &sza_down,
                      &eclipse_entry, &eclipse_exit,
                      &sun_moon_entry, &sun_moon_exit,
                      ierr);

    /* Or, using the run_id */
    long run_id;

    status= xv_orbit_extra_run (&run_id, &orbit, orbit_info_vector,
                      &num_sza, sza,
                      &sza_up, &sza_down,
                      &eclipse_entry, &eclipse_exit,
                      &sun_moon_entry, &sun_moon_exit,
                      ierr);
}
```

7.13.3 Input parameters xv_orbit_extra

Table 43: Input parameters of xv_orbit_extra

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit _id*	-	Structure that contains the orbit data	-	-
orbit	long	-	absolute orbit number		≥ start osf
orbit_info_vector [XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS]	double	[0]	repeat_cycle	days	>0
		[1]	cycle_length	orbits	>0
		[2]	MLST drift		s/day
		[3]	MLST	deg	> 0 <360
		[4]	phasing	deg	> 0 <360
		[5]	UTC time at ascending node	days (processing format)	
		[6-8]	position at ANX	m	
		[9-11]	velocity at ANX	m/s	
		[12-17]	mean keplerian elements at ANX		
		[18-23]	osculating keplerian elements at ANX		
		[24]	Nodal period	s	
num_sza	long	-	Number of Sun Zenit angles in the sza array	-	>0
sza	double*	all	list of Sun Zenit angles to compute	deg	≥ 0 ≤ 180

7.13.4 Output parameters xv_orbit_extra

Table 44: Output parameters of xv_orbi_extra

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_orbit_extra	long	-	Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
sza_up	double	all	Seconds since ANX of Sun Zenith Angles when SZA is increasing with time.	s	≥ 0 ≤ orb. period
sza_down	double	all	Seconds since ANX of Sun Zenith Angles when SZA is decreasing with time.	s	≥ 0 ≤orb. period
eclipse_entry	double	-	Seconds since ANX of eclipse entry. Note that the value is provided within the input orbit, so that the eclipse_exit will be less than the eclipse_entry if the ANX is in eclipse.	s	≥ 0 ≤ orbital period -1 if there is not eclipse
eclipse_exit	double	-	Seconds since ANX of eclipse exit. Note that the value is provided within the input orbit, so that the eclipse_exit will be less than the eclipse_entry if the ANX is in eclipse.	s	≥ 0 ≤ orbital period -1 if there is not eclipse
sun_moon_entry	double	-	Seconds since ANX of Sun Occultation by Moon entry.	s	<-1 if no occultation is found ≥ 0 ≤ orbital period
sun_moon_exit	double	-	Seconds since ANX of Sun Occultation by Moon exit	s	<-1 if no occultation is found ≥ 0 ≤ orbital period
ierr	long*		Error status flags		

<u>Note (Memory Management):</u> Note that the sza_up and sza_down arrays are pointers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_orbit_extra** function. So the user

will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.13.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_orbit_extra** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_orbit_extra** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 45: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input orbit Id.	Computation not performed	XV_CFI_ORBIT_EXTRA_ORBIT_STATUS_ER R	0
ERR	Error allocating memory for SZA entry/exit times	Computation not performed	XV_CFI_ORBIT_EXTRA_MEM_ERR	1
ERR	Error computing SZA entry/exit times	Computation not performed	XV_CFI_ECLIPSE_XL_EF_TO_QEF_ERR	2
ERR	Error computing eclipse entry/exit times	Computation not performed	XV_CFI_ORBIT_EXTRA_ECLIPSE_ERR	3
ERR	Error computing Sun occultation by Moon.	Computation not performed	XV_CFI_ORBIT_EXTRA_SUN_OCC_BY_MOON_ERR	4

7.14 xv_gps_vis_time

TBW

7.15 xv_time_segments_not

7.15.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_not** instead.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_time_segments_not** function computes the compliment of a list of orbital segments (see Figure 22)

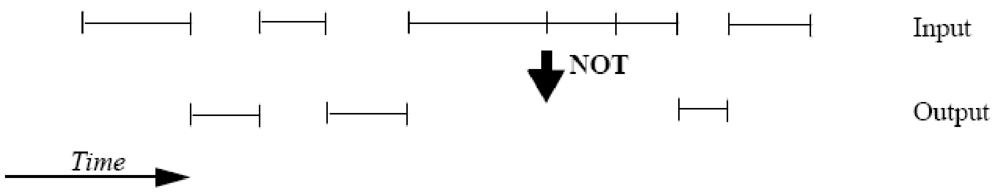


Figure 22: xv_time_segment_not_function

Note that the intervals from the first orbit to the first segment and from the last segment to the end of mission are not returned.

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals used by **xv_time_segments_not** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.
- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of micro seconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

The **xv_time_segments_not** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.15.2 Calling sequence xv_time_segments_not

For C programs, the call to **xv_time_segments_not** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

       xo_orbit_id orbit_id = {NULL};
       long     orbit_type, order_switch,
                num_segments_in,
                *bgn_orbit_in, *bgn_secs_in,
                *bgn_microsecs_in, *bgn_cycle_in,
                *end_orbit_in, *end_secs_in,
                *end_microsecs_in, *end_cycle_in,
                num_segments_out,
                *bgn_orbit_out, *bgn_secs_out,
                *bgn_microsecs_out, *bgn_cycle_out,
                *end_orbit_out, *end_secs_out,
                *end_microsecs_out, *end_cycle_out,
                ierr[XV_NUM_ERR_NOT], status;

       status = xv_time_segments_not(
                         &orbit_id,
                         &orbit_type,  &order_switch,
                         &number_segments_in,
                         bgn_orbit_in, bgn_secs_in,
                         bgn_microsecs_in, bgn_cycle_in,
                         end_orbit_in, end_secs_in,
                         end_microsecs_in, end_cycle_in,
                         &num_segments_out,
                         &bgn_orbit_out, &bgn_secs_out,
                         &bgn_microsecs_out, &bgn_cycle_out,
                         &end_orbit_out, &end_secs_out,
                         &end_microsecs_out, &end_cycle_out,
                         ierr);


       /* Or, using the run_id */
       long run_id;
```

```
        status = xv_time_segments_not_run(
                                  &run_id,
                                  &orbit_type,  &order_switch,
                                  &number_segments_in,
                                  bgn_orbit_in, bgn_secs_in,
                                  bgn_microsecs_in, bgn_cycle_in,
                                  end_orbit_in, end_secs_in,
                                  end_microsecs_in, end_cycle_in,
                                  &num_segments_out,
                                  &bgn_orbit_out, &bgn_secs_out,
                                  &bgn_microsecs_out, &bgn_cycle_out,
                                  &end_orbit_out, &end_secs_out,
                                  &end_microsecs_out, &end_cycle_out,
                                  ierr);
    }
```

7.15.3 Input parameters xv_time_segments_not

Table 46: Input parameters of xv_time_segments_not

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
num_segments_in	long	-	Number of segments in the input list.	-	>0
bgn_orbit_in	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs_in	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs_in	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle_in	long*	all	Array of cycle numbers for the beginning of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit_in	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs_in	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs_in	long*	all	Array of seconds within a second for the end of the segments	-	>0 <999999
end_cycle_in	long*	all	Array of cycle numbers for the end of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL

7.15.4 Output parameters xv_time_segments_not

Table 47: Output parameters of xv_time_segments_not

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_not	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
num_segments_out	long	-	Number of segments in the output list.	-	>0
bgn_orbit_out	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs_out	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs_out	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 < 999999
bgn_cycle_out	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit_out	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs_out	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs_out	long*	all	Array of microseconds within a second for the end of the segments	-	>0 < 999999
end_cycle_out	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
ierr[10]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_time_segments_not** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.15.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_not** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_not** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 48: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_ NOT_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_REL_TO_ABS_ORBIT _ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_ABS_TO_REL_ORBIT _ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_ NOT_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_GEO_SAT_ERR	4
WARN	Deprecated function. Use xv_timesegments_compute_not instead	Computation performed	XV_CFI_TIME_SEGMENTS_NOT_DEPRECATED_WARN	5

7.16 xv_timesegments_compute_not

7.16.1 Overview

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_timesegments_compute_not** function computes the compliment of a list of orbital segments (see Figure 23)

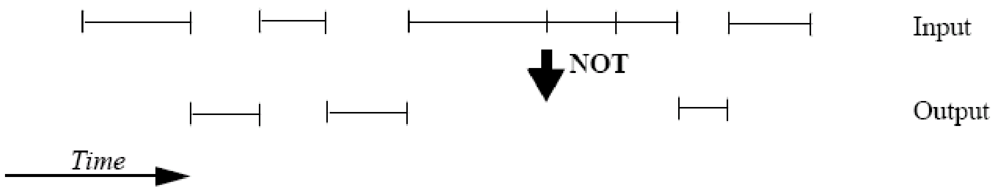


Figure 23: xv_timesegments_compute_not function

Note that the intervals from the first orbit to the first segment and from the last segment to the end of mission are not returned.

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals (xv_time_interval) used by **xv_timesegments_compute_not** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times. Moreover, the segments will be ordered chronologically.

The **xv_timesegments_compute_not** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.16.2 Calling sequence xv_timesegments_compute_not

For C programs, the call to **xv_timesegments_compute_not** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

        xo_orbit_id orbit_id = {NULL};
        long     order_switch;
        xv_visibility_interval_list seg_in;
        xv_visibility_interval_list seg_out;
        long ierr[XV_NUM_ERR_COMPUTE_NOT];
        long status;

        status = xv_timesegments_compute_not(
                        &orbit_id, &order_switch,
                        &seg_in,
                        &seg_out, ierr);

        /* Or, using the run_id */
        long run_id;

        status = xv_timesegments_compute_not(
                        &run_id, &order_switch,
                        &seg_in,
                        &seg_out, ierr);
```

7.16.3 Input parameters xv_timesegments_compute_not

Table 49: Input parameters of xv_timesegments_compute_not

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
seg_in	xv_visibility_interval_list	-	Input list of segments	-	-

7.16.4 Output parameters xv_timesegments_compute_not

Table 50: Output parameters of xv_timesegments_compute_not

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_timesegments_compute_not	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
seg_out	xv_visibility_interval_list	-	Output list of segments	-	-
ierr[]	long		Error status flags		

Memory Management: Note that the memory for the output visibility segments arrays is allocated within the **xv_timesegments_compute_not** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.16.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_not** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_not** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 51: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_REL_TO_ABS_ORBIT _ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_ABS_TO_REL_ORBIT _ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_GEO_SAT_ERR	4
ERR	Error computing time to orbit parameters	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_TIME_TO_ORBIT_ERR	5
ERR	Error computing UTC time for orbit	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_NOT_GET_UTC_TIME_ERR	6

7.17 xv_time_segments_or

7.17.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_or** instead.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_time_segments_or** function computes the union of a list of orbital segments (see Figure 24)

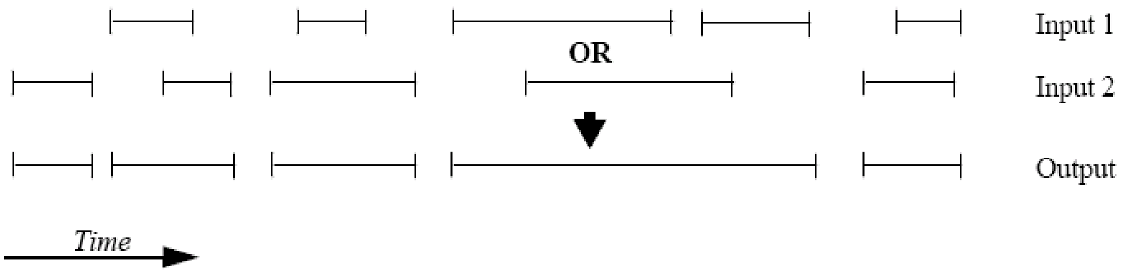


Figure 24: xv_time_segments_or_function

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals used by **xv_time_segments_or** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.
- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

The **xv_time_segments_or** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.17.2 Calling sequence xv_time_segments_or

For C programs, the call to **xv_time_segments_or** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

      xo_orbit_id orbit_id = {NULL};
      long      orbit_type, order_switch,
                num_segments_1,
                *bgn_orbit_1, *bgn_secs_1,
                *bgn_microsecs_1, *bgn_cycle_1,
                *end_orbit_1, *end_secs_1,
                *end_microsecs_1, *end_cycle_1,
                num_segments_2,
                *bgn_orbit_2, *bgn_secs_2,
                *bgn_microsecs_2, *bgn_cycle_2,
                *end_orbit_2, *end_secs_2,
                *end_microsecs_2, *end_cycle_2,
                num_segments_out,
                *bgn_orbit_out, *bgn_secs_out,
                *bgn_microsecs_out, *bgn_cycle_out,
                *end_orbit_out, *end_secs_out,
                *end_microsecs_out, *end_cycle_out,
                ierr[XV_NUM_ERR_OR], status;

      status = xv_time_segments_or (
                         &orbit_id,
                         &orbit_type, &order_switch,
                         &number_segments_1,
                         bgn_orbit_1, bgn_second_1,
                         bgn_microsec_1, bgn_cycle_1,
                         end_orbit_1, end_second_1,
                         end_microsec_1, end_cycle_1,
                         &number_segments_2,
                         bgn_orbit_2, bgn_second_2,
                         bgn_microsec_2, bgn_cycle_2,
                         end_orbit_2, end_second_2,
                         end_microsec_2, end_cycle_2,
                         &num_segments_out,
```

```
                              &bgn_orbit_out, &bgn_secs_out,
                              &bgn_microsecs_out, &bgn_cycle_out,
                              &end_orbit_out, &end_secs_out,
                              &end_microsecs_out, &end_cycle_out,
                              ierr);




        /* Or, using the run_id */
        long run_id;

        status = xv_time_segments_or_run (
                             &run_id,
                              &orbit_type, &order_switch,
                              &number_segments_1,
                              bgn_orbit_1, bgn_second_1,
                              bgn_microsec_1, bgn_cycle_1,
                              end_orbit_1, end_second_1,
                              end_microsec_1, end_cycle_1,
                              &number_segments_2,
                              bgn_orbit_2, bgn_second_2,
                              bgn_microsec_2, bgn_cycle_2,
                              end_orbit_2, end_second_2,
                              end_microsec_2, end_cycle_2,
                              &num_segments_out,
                              &bgn_orbit_out, &bgn_secs_out,
                             &bgn_microsecs_out, &bgn_cycle_out,
                              &end_orbit_out, &end_secs_out,
                             &end_microsecs_out, &end_cycle_out,
                             ierr);

    }
```

7.17.3 Input parameters xv_time_segments_or

Table 52: Input parameters of xv_time_segments_or

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
num_segments_1	long	-	Number of segments in the input list 1.	-	>0
bgn_orbit_1	long*	all	Array of orbit numbers for the beginning of the segments in list 1	-	>0
bgn_secs_1	long*	all	Array of seconds elapsed since ANX for the beginning of the segments in list 1	-	>0 <nodal period
bgn_microsecs_1	long*	all	Array of microseconds within a second for the beginning of the segments in list 1	-	>0 <999999
bgn_cycle_1	long*	all	Array of cycle numbers for the beginning of the segments in list 1. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit_1	long*	all	Array of orbit numbers for the end of the segments in list 1	-	>0
end_secs_1	long*	all	Array of seconds elapsed since ANX for the end of the segments in list 1	-	>0 <nodal period
end_microsecs_1	long*	all	Array of microseconds within a second for the end of the segments in list 1	-	>0 <999999
end_cycle_1	long*	all	Array of cycle numbers for the end of the segments in list 1. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
num_segments_2	long	-	Number of segments in the input list 2.	-	>0

bgn_orbit_2	long*	all	Array of orbit numbers for the beginning of the segments in list 2	-	>0
bgn_secs_2	long*	all	Array of seconds elapsed since ANX for the beginning of the segments in list 2	-	>0 <nodal period
bgn_microsecs_2	long*	all	Array of microseconds within a second for the beginning of the segments in list 2	-	>0 <999999
bgn_cycle_2	long*	all	Array of cycle numbers for the beginning of the segments in list 2. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit_2	long*	all	Array of orbit numbers for the end of the segments in list 2	-	>0
end_secs_2	long*	all	Array of seconds elapsed since ANX for the end of the segments in list 2	-	>0 <nodal period
end_microsecs_2	long*	all	Array of microseconds within a second for the end of the segments in list 2	-	>0 <999999
end_cycle_2	long*	all	Array of cycle numbers for the end of the segments in list 2. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL

7.17.4 Output parameters xv_time_segments_or

Table 53: Output parameters of xv_time_segments_or

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_or	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
num_segments_out	long	-	Number of segments in the output list.	-	>0
bgn_orbit_out	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs_out	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs_out	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle_out	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit_out	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs_out	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs_out	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle_out	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
ierr[10]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_time_segments_or** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.17.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_or** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_or** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 54: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_OR_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_OR_REL_TO_ABS_ORBIT_ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_OR_ABS_TO_REL_ORBIT_ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_OR_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_OR_GEO_SAT_ERR	4
WARN	Deprecated function. Use xv_timesegments_compute_or instead	Computation performed	XV_CFI_TIME_SEGMENTS_OR_DEPRECATED_WARN	5

7.18 xv_timesegments_compute_or

7.18.1 Overview

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_timesegments_compute_or** function computes the union of a list of orbital segments (see Figure 25)

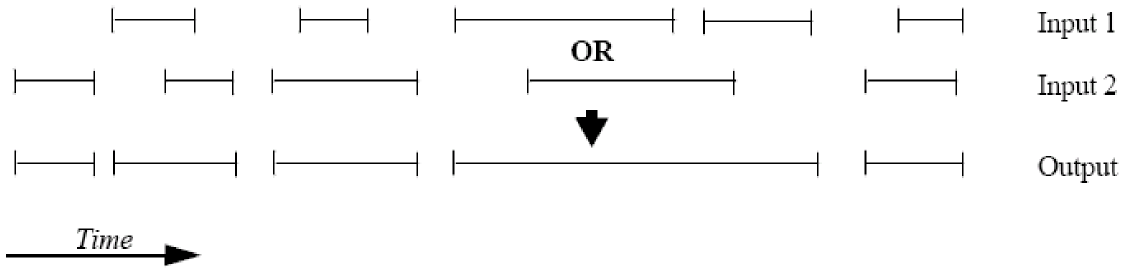


Figure 25: xv_timesegments_compute_or_function

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals (xv_time_interval) used by **xv_timesegments_compute_or** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times. Moreover, the segments will be ordered chronologically.

The **xv_timesegments_compute_or** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.18.2 Calling sequence xv_timesegments_compute_or

For C programs, the call to **xv_timesegments_compute_or** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

        xo_orbit_id orbit_id = {NULL};
        long     order_switch;
        xv_visibility_interval_list seg_in1, seg_in2;
        xv_visibility_interval_list seg_out;
        long ierr[XV_NUM_ERR_COMPUTE_NOT];
        long status;

        status = xv_timesegments_compute_not(
                         &orbit_id, &order_switch,
                         &seg_in1, 6seg_in2
                         &seg_out, ierr);


        /* Or, using the run_id */
        long run_id;

        status = xv_timesegments_compute_not(
                         &run_id, &order_switch,
                         &seg_in1, &seg_in2
                         &seg_out, ierr);

}
```

7.18.3 Input parameters xv_timesegments_compute_or

Table 55: Input parameters of xv_timesegments_compute_or

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
seg_in1	xv_visibility_interval_list	-	Input list of segments 1	-	-
seg_in2	xv_visibility_interval_list	-	Input list of segments 2	-	-

7.18.4 Output parameters xv_timesegments_compute_or

Table 56: Output parameters of xv_timesegments_compute_or

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_timesegments_compute_or	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
seg_out	xv_visibility_interval_list	-	Output list of segments	-	-
ierr[]	long		Error status flags		

Memory Management: Note that the memory for the output visibility segments arrays is allocated within the **xv_timesegments_compute_or** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.18.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_or** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_or** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 57: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_REL_TO_ABS_ORBIT_ ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_ABS_TO_REL_ORBIT_ ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_GEO_SAT_ERR	4
ERR	Wrong input time type	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_TIME_TYPE_ERR	5
ERR	Error computing time to orbit parameters	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_TIME_TO_ORBIT_ERR	6
ERR	Error computing UTC time for orbit	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_OR_GET_UTC_TIME_ERR	7

7.19 xv_time_segments_and

7.19.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_and** instead.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_time_segments_and** function computes the intersection of a list of orbital segments (see Figure 26)

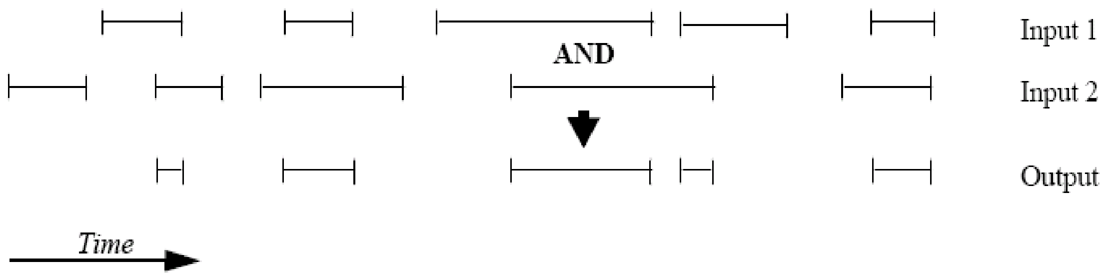


Figure 26: xv_time_segments_and_function

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals used by **xv_time_segments_and** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.
- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

The **xv_time_segments_and** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.19.2 Calling sequence xv_time_segments_and

For C programs, the call to **xv_time_segments_and** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

      xo_orbit_id   orbit_id = {NULL};
      long     orbit_type, order_switch,
               num_segments_1,
               *bgn_orbit_1, *bgn_secs_1,
               *bgn_microsecs_1, *bgn_cycle_1,
               *end_orbit_1, *end_secs_1,
               *end_microsecs_1, *end_cycle_1,
               num_segments_2,
               *bgn_orbit_2, *bgn_secs_2,
               *bgn_microsecs_2, *bgn_cycle_2,
               *end_orbit_2, *end_secs_2,
               *end_microsecs_2, *end_cycle_2,
               num_segments_out,
               *bgn_orbit_out, *bgn_secs_out,
               *bgn_microsecs_out, *bgn_cycle_out,
               *end_orbit_out, *end_secs_out,
               *end_microsecs_out, *end_cycle_out,
               ierr[XV_NUM_ERR_AND], status;

      status = xv_time_segments_and (
                         &orbit_id,
                         &orbit_type, &order_switch,
                         &number_segments_1,
                         bgn_orbit_1, bgn_second_1,
                         bgn_microsec_1, bgn_cycle_1,
                         end_orbit_1, end_second_1,
                         end_microsec_1, end_cycle_1,
                         &number_segments_2,
                         bgn_orbit_2, bgn_second_2,
                         bgn_microsec_2, bgn_cycle_2,
                         end_orbit_2, end_second_2,
                         end_microsec_2, end_cycle_2,
                         &num_segments_out,
```

```
                              &bgn_orbit_out, &bgn_secs_out,
                              &bgn_microsecs_out, &bgn_cycle_out,
                              &end_orbit_out, &end_secs_out,
                              &end_microsecs_out, &end_cycle_out,
                              ierr);



        /* Or, using the run_id */
        long run_id;

        status = xv_time_segments_and_run (
                              &run_id,
                              &orbit_type, &order_switch,
                              &number_segments_1,
                              bgn_orbit_1, bgn_second_1,
                              bgn_microsec_1, bgn_cycle_1,
                              end_orbit_1, end_second_1,
                              end_microsec_1, end_cycle_1,
                              &number_segments_2,
                              bgn_orbit_2, bgn_second_2,
                              bgn_microsec_2, bgn_cycle_2,
                              end_orbit_2, end_second_2,
                              end_microsec_2, end_cycle_2,
                              &num_segments_out,
                              &bgn_orbit_out, &bgn_secs_out,
                              &bgn_microsecs_out, &bgn_cycle_out,
                              &end_orbit_out, &end_secs_out,
                              &end_microsecs_out, &end_cycle_out,
                              ierr);

    }
```

7.19.3 Input parameters xv_time_segments_and

Table 58: Input parameters of xv_time_segments_and

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
num_segments_1	long	-	Number of segments in the input list 1.	-	>0
bgn_orbit_1	long*	all	Array of orbit numbers for the beginning of the segments in list 1	-	>0
bgn_secs_1	long*	all	Array of seconds elapsed since ANX for the beginning of the segments in list 1	-	>0 <nodal period
bgn_microsecs_1	long*	all	Array of microseconds within a second for the beginning of the segments in list 1	-	>0 <999999
bgn_cycle_1	long*	all	Array of cycle numbers for the beginning of the segments in list 1. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit_1	long*	all	Array of orbit numbers for the end of the segments in list 1	-	>0
end_secs_1	long*	all	Array of seconds elapsed since ANX for the end of the segments in list 1	-	>0 <nodal period
end_microsecs_1	long*	all	Array of microseconds within a second for the end of the segments in list 1	-	>0 <999999
end_cycle_1	long*	all	Array of cycle numbers for the end of the segments in list 1. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
num_segments_2	long	-	Number of segments in the input list 2.	-	>0

bgn_orbit_2	long*	all	Array of orbit numbers for the beginning of the segments in list 2	-	>0
bgn_secs_2	long*	all	Array of seconds elapsed since ANX for the beginning of the segments in list 2	-	>0 <nodal period
bgn_microsecs_2	long*	all	Array of microseconds within a second for the beginning of the segments in list 2	-	>0 <999999
bgn_cycle_2	long*	all	Array of cycle numbers for the beginning of the segments in list 2. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit_2	long*	all	Array of orbit numbers for the end of the segments in list 2	-	>0
end_secs_2	long*	all	Array of seconds elapsed since ANX for the end of the segments in list 2	-	>0 <nodal period
end_microsecs_2	long*	all	Array of microseconds within a second for the end of the segments in list 2	-	>0 <999999
end_cycle_2	long*	all	Array of cycle numbers for the end of the segments in list 2. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL

7.19.4 Output parameters xv_time_segments_and

Table 59: Output parameters of xv_time_segments_and

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_and	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
num_segments_out	long	-	Number of segments in the output list.	-	>0
bgn_orbit_out	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs_out	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs_out	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle_out	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit_out	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs_out	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs_out	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle_out	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
ierr[10]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_time_segments_and** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.19.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_and** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_and** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 60: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_REL_TO_ABS_ORBIT_ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_ABS_TO_REL_ORBIT_ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_GEO_SAT_ERR	4
WARN	Deprecated function. Use xv_timesegments_compute_and instead	Computation performed	XV_CFI_TIME_SEGMENTS_AND_DEPRECATED_WARN	5

7.20 xv_timesegments_compute_and

7.20.1 Overview

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_timesegments_compute_and** function computes the intersection of a list of orbital segments (see Figure 27)

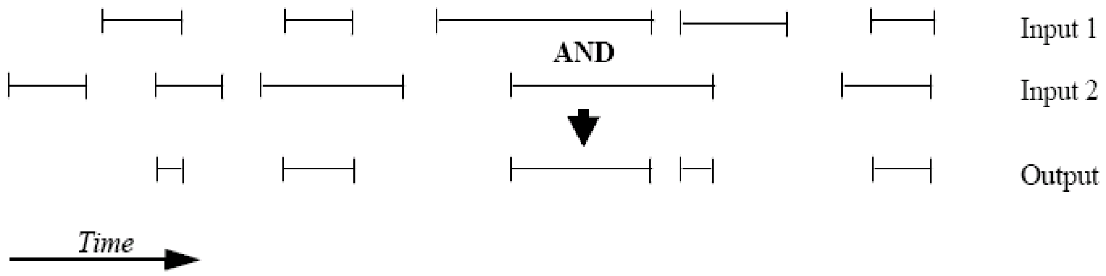


Figure 27: xv_timesegments_compute_and_function

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals (xv_time_interval) used by **xv_timesegments_compute_and** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times. Moreover, the segments will be ordered chronologically.

The **xv_timesegments_compute_and** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.20.2 Calling sequence xv_timesegments_compute_and

For C programs, the call to **xv_timesegments_compute_and** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

        xo_orbit_id orbit_id = {NULL};
        long     order_switch;
        xv_visibility_interval_list seg_in1, seg_in2;
        xv_visibility_interval_list seg_out;
        long ierr[XV_NUM_ERR_COMPUTE_AND];
        long status;

        status = xv_timesegments_compute_and(
                            &orbit_id, &order_switch,
                            &seg_in1, &seg_in2,
                            &seg_out, ierr);


        /* Or, using the run_id */
        long run_id;

        status = xv_timesegments_compute_and(
                            &run_id, &order_switch,
                            &seg_in1, &seg_in2,
                            &seg_out, ierr);

}
```

7.20.3 Input parameters xv_timesegments_compute_and

Table 61: Input parameters of xv_timesegments_compute_and

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
seg_in1	xv_visibility_interval_list	-	Input list of segments 1	-	-
seg_in2	xv_visibility_interval_list	-	Input list of segments 2	-	-

7.20.4 Output parameters xv_timesegments_compute_and

Table 62: Output parameters of xv_timesegments_compute_and

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_timesegments_compute_and	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
seg_out	xv_visibility_interval_list	-	Output list of segments	-	-
ierr[]	long		Error status flags		

Memory Management: Note that the memory for the output visibility segments arrays is allocated within the **xv_timesegments_compute_and** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.20.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_and** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_and** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 63: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_REL_TO_ABS_ORBIT _ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_ABS_TO_REL_ORBIT _ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_GEO_SAT_ERR	4
ERR	Wrong input time type	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_TIME_TYPE_ERR	5
ERR	Error computing time to orbit parameters	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_TIME_TO_ORBIT_ERR	6
ERR	Error computing UTC time for orbit	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_AND_GET_UTC_TIME_ERR	7

7.21 xv_time_segments_sort

7.21.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_sort** instead.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_time_segments_sort** function sorts a list of orbital segments following two different criteria:

- Absolute orbits: the segments are sorted by their start time
- Relative orbits

The time intervals used by **xv_time_segments_sort** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.
- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Note that the sort criteria does not have any relation with the chosen orbit representation. The following example clarifies this:

Input orbits: 6, 8, 4, 5, 9, 3 (absolute)
Let's suppose that the cycle length is 4 orbits. Then the relative orbits are:
input orbits: 2, 4, 4, 1, 1, 3 (relative)

When ordering this array, we have the following possibilities (Table 64) depending on the orbit representation and the sort criteria chosen:

Table 64: xv_time_segments_sort function

Input	Sort Criteria	Output
absolute orbits 6, 8, 4, 5, 9, 3	absolute orbits	absolute orbits 3, 4, 5, 6, 8, 9
	relative orbits	absolute orbits 5, 9, 6, 3, 4, 8
relative orbits 2, 4, 4, 1, 1, 3	absolute orbits	relative orbits 3, 4, 1, 2, 4, 1
	relative orbits	relative orbits 1, 1, 2, 3, 4, 4

The **xv_time_segments_sort** requires access the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.21.2 Calling sequence xv_time_segments_sort

For C programs, the call to **xv_time_segments_sort** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

      xo_orbit_id   orbit_id = {NULL};
      long     orbit_type, sort_criteria,
               num_segments,
               *bgn_orbit, *bgn_secs,
               *bgn_microsecs, *bgn_cycle,
               *end_orbit, *end_secs,
               *end_microsecs, *end_cycle,
               ierr, status;

      status = xv_time_segments_sort (
                        &orbit_id,
                        &orbit_type, &sort_criteria,
                        &number_segments,
                        bgn_orbit, bgn_second,
                        bgn_microsec, bgn_cycle,
                        end_orbit, end_second,
                        end_microsec, end_cycle,
                        ierr);

      /* Or, using the run_id */
      long run_id;

      status = xv_time_segments_sort_run (
                        &run_id,
                        &orbit_type, &sort_criteria,
                        &number_segments,
                        bgn_orbit, bgn_second,
                        bgn_microsec, bgn_cycle,
                        end_orbit, end_second,
                        end_microsec, end_cycle,
                        ierr);
}
```

7.21.3 Input parameters xv_time_segments_sort

Table 65: Input parameters of xv_time_segments_sort

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
sort_criteria	long	-	sorting criteria to be used: absolute or relative orbits	-	Complete (see Table 3)
num_segments	long	-	Number of segments in the input.	-	>0
bgn_orbit	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle	long*	all	Array of cycle numbers for the beginning of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs	long*	all	Array of microseconds within a second for the end of the segments.	-	>0 <999999
end_cycle	long*	all	Array of cycle numbers for the end of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL

7.21.4 Output parameters xv_time_segments_sort

Table 66: Output parameters of xv_time_segments_sort

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_and	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
ierr[10]	long		Error status flags		

7.21.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_sort** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_sort** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 67: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_ SORT_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_SORT_CHANGING_ORBIT_ ERR	1
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_SORT_GEO_SAT_ERR	2
WARN	Deprecated function. Use xv_timesegments_compute_sort instead	Computation performed	XV_CFI_TIME_SEGMENTS_SORT_DEPRECATED_WARN	3

7.22 xv_timesegments_compute_sort

7.22.1 Overview

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_timesegments_compute_sort** function sorts a list of orbital segments following two different criteria:

- Absolute orbits: the segments are sorted by their start time
- Relative orbits

The time intervals (xv_time_interval) used by **xv_timesegments_compute_and** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times.

Note that the sort criteria does not have any relation with the chosen orbit representation. The following example clarifies this:

Input orbits: 6, 8, 4, 5, 9, 3 (absolute)
Let's suppose that the cycle length is 4 orbits. Then the relative orbits are:
input orbits: 2, 4, 4, 1, 1, 3 (relative)

When ordering this array, we have the following possibilities (Table 68) depending on the orbit representation and the sort criteria chosen:

Table 68: xv_timesegments_compute_sort function

Input	Sort Criteria	Output
absolute orbits 6, 8, 4, 5, 9, 3	absolute orbits	absolute orbits 3, 4, 5, 6, 8, 9
	relative orbits	absolute orbits 5, 9, 6, 3, 4, 8
relative orbits 2, 4, 4, 1, 1, 3	absolute orbits	relative orbits 3, 4, 1, 2, 4, 1
	relative orbits	relative orbits 1, 1, 2, 3, 4, 4

The **xv_timesegments_compute_sort** requires access the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.22.2 Calling sequence xv_timesegments_compute_sort

For C programs, the call to **xv_timesegments_compute_sort** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

       xo_orbit_id orbit_id = {NULL};
       long     sort_criteria;
       xv_visibility_interval_list seg_in;
       long ierr[XV_NUM_ERR_COMPUTE_SORT];
       long status;

       status = xv_timesegments_compute_sort(
                         &orbit_id, &sort_criteria,
                         &seg_in, ierr);


       /* Or, using the run_id */
       long run_id;

       status = xv_timesegments_compute_sort(
                         &run_id, &sort_criteria,
                         &seg_in, ierr);
}
```

7.22.3 Input parameters xv_timesegments_compute_sort

Table 69: Input parameters of xv_timesegments_compute_sort

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
sort_criteria	long	-	sorting criteria to be used: absolute or relative orbits	-	Complete (see Table 3)
seg_in	xv_visibility_interval_list	-	Input list of segments	-	-

7.22.4 Output parameters xv_timesegments_compute_sort

Table 70: Output parameters of xv_timesegments_compute_sort

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_sort	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
ierr[]	long		Error status flags		

7.22.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_sort** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_sort** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 71: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_SORT_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_SORT_CHANGING_ORBIT_ ERR	1
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_SORT_GEO_SAT_ERR	2
ERR	Wrong input time type	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_SORT_TIME_TYPE_ERR	3
ERR	Error computing time to orbit parameters	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_SORT_TIME_TO_ORBIT_ERR	4
ERR	Error computing UTC time for orbit	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_SORT_GET_UTC_TIME_ERR	5

7.23 xv_time_segments_merge

7.23.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_merge** instead.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_time_segments_merge** function merges all the overlapped segments within a list (see Figure 28)

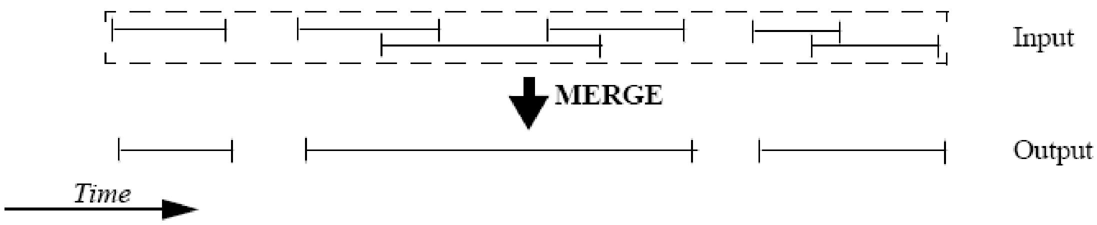


Figure 28: xv_time_segments_merge function

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals used by **xv_time_segments_merge** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.
- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. Moreover, the segments will be ordered chronologically.

The **xv_time_segments_merge** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.23.2 Calling sequence xv_time_segments_merge

For C programs, the call to **xv_time_segments_merge** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

      xo_orbit_id   orbit_id = {NULL};
      long     orbit_type, order_switch,
               num_segments,
               *bgn_orbit, *bgn_secs,
               *bgn_microsecs, *bgn_cycle,
               *end_orbit, *end_secs,
               *end_microsecs, *end_cycle,
               num_segments_out,
               *bgn_orbit_out, *bgn_secs_out,
               *bgn_microsecs_out, *bgn_cycle_out,
               *end_orbit_out, *end_secs_out,
               *end_microsecs_out, *end_cycle_out,
               ierr[XV_NUM_ERR_MERGE], status;

      status = xv_time_segments_merge(
                         &orbit_id,
                         &orbit_type,  &order_switch,
                         &number_segments,
                         bgn_orbit, bgn_secs,
                         bgn_microsecs, bgn_cycle,
                         end_orbit, end_secs,
                         end_microsecs, end_cycle,
                         &num_segments_out,
                         &bgn_orbit_out, &bgn_secs_out,
                         &bgn_microsecs_out, &bgn_cycle_out,
                         &end_orbit_out, &end_secs_out,
                         &end_microsecs_out, &end_cycle_out,
                         ierr);

      /* Or, using the run_id */
      long run_id;

      status = xv_time_segments_merge_run(
```

```
                                    &run_id,
                                    &orbit_type,  &order_switch,
                                    &number_segments,
                                    bgn_orbit, bgn_secs,
                                    bgn_microsecs, bgn_cycle,
                                    end_orbit, end_secs,
                                    end_microsecs, end_cycle,
                                    &num_segments_out,
                                    &bgn_orbit_out, &bgn_secs_out,
                                    &bgn_microsecs_out, &bgn_cycle_out,
                                    &end_orbit_out, &end_secs_out,
                                    &end_microsecs_out, &end_cycle_out,
                                    ierr);
     }
```

7.23.3 Input parameters xv_time_segments_merge

Table 72: Input parameters of xv_time_segments_merge

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
num_segments_in	long	-	Number of segments in the input list.	-	>0
bgn_orbit	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle	long*	all	Array of cycle numbers for the beginning of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle	long*	all	Array of cycle numbers for the end of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL

7.23.4 Output parameters xv_time_segments_merge

Table 73: Output parameters of xv_time_segments_merge

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_merge	long		Function status flag, = 0 No error > 0Warnings, results generated < 0Error, no results generated		
num_segments_out	long	-	Number of segments in the output list.	-	>0
bgn_orbit_out	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs_out	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs_out	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle_out	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit_out	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs_out	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs_out	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle_out	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
ierr[10]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_time_segments_merge** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.23.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_merge** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_merge** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 74: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_ MERGE_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_MERGE_REL_TO_ABS_OR BIT_ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_MERGE_ABS_TO_REL_OR BIT_ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_ MERGE_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MERGE_GEO_SAT_ERR	4
WARN	Deprecated function. Use xv_timesegments_compute_merge instead	Computation performed	XV_CFI_TIME_SEGMENTS_MERGE_DEPRECATED_WARN	5

7.24 xv_timesegments_compute_merge

7.24.1 Overview

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_timesegments_compute_merge** function merges all the overlapped segments within a list (see Figure 29)

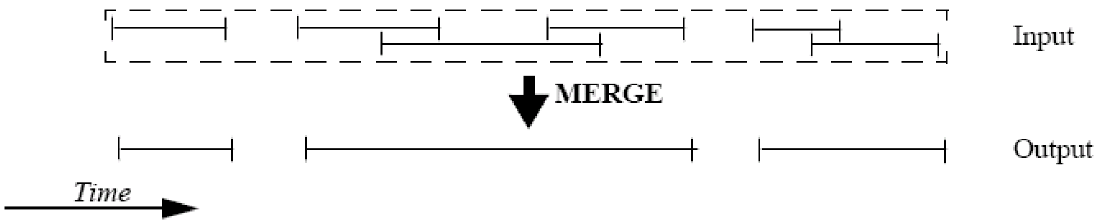


Figure 29: xv_timesegments_compute_merge function

The input segments list need to be sorted according to the start time of the segments. If this list is not sorted, it should be indicated in the function interface with the corresponding parameter (see below). In this case the input list will be modified accordingly.

The time intervals (xv_time_interval) used by **xv_timesegments_compute_and** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times. Moreover, the segments will be ordered chronologically.

The **xv_timesegments_compute_merge** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.24.2 Calling sequence xv_timesegments_compute_merge

For C programs, the call to **xv_timesegments_compute_merge** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{

      xo_orbit_id orbit_id = {NULL};
      long     order_switch;
      xv_visibility_interval_list seg_in1, seg_in2;
      xv_visibility_interval_list seg_out;
      long ierr[XV_NUM_ERR_COMPUTE_MERGE];
      long status;

      status = xv_timesegments_compute_merge(
                         &orbit_id, &order_switch,
                         &seg_in,
                         &seg_out, ierr);


      /* Or, using the run_id */
      long run_id;

      status = xv_timesegments_compute_merge(
                         &run_id, &order_switch,
                         &seg_in,
                         &seg_out, ierr);
}
```

7.24.3 Input parameters xv_timesegments_compute_merge

Table 75: Input parameters of xv_timesegments_compute_merge

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
order_switch	long	-	Indicates if the input list is sorted by start times. If input segments are already sorted, the flag should be set to XV_TIME_ORDER to save computation time.	-	Complete (see Table 3)
seg_in	xv_visibility_interval_list	-	Input list of segments	-	-

7.24.4 Output parameters xv_timesegments_compute_merge

Table 76: Output parameters of xv_timesegments_compute_merge

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_timesegments_compute_merge	long		Function status flag, = 0 No error > 0Warnings, results generated < 0Error, no results generated		
seg_out	xv_visibility_interval_list	-	Output list of segments	-	-
ierr[]	long		Error status flags		

Memory Management: Note that the memory for the output visibility segments arrays is allocated within the **xv_timesegments_compute_merge** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.24.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_merge** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_merge** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 77: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_REL_TO_ABS_OR BIT_ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_ABS_TO_REL_OR BIT_ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_GEO_SAT_ERR	4
ERR	Wrong input time type	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_TIME_TYPE_ERR	5
ERR	Error computing time to orbit parameters	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_TIME_TO_ORBIT_ERR	6
ERR	Error computing UTC time for orbit	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MERGE_GET_UTC_TIME_ERR	7

7.25 xv_time_segments_delta

7.25.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_delta** instead.

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_time_segments_delta** function makes all the segments within a list, longer or shorter. After increasing/decreasing the longitude of the segments, these are sorted and merged to avoid possible overlapping. Therefore, at the end the list is sorted and without overlapped segments.

The time intervals used by **xv_time_segments_delta** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.
- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits.

The **xv_time_segments_delta** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.25.2 Calling sequence xv_time_segments_delta

For C programs, the call to **xv_time_segments_delta** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
      xo_orbit_id   orbit_id = {NULL};
      long      orbit_type,
                num_segments,
                *bgn_orbit, *bgn_secs,
                *bgn_microsecs, *bgn_cycle,
                *end_orbit, *end_secs,
                *end_microsecs, *end_cycle,
                num_segments_out,
                *bgn_orbit_out, *bgn_secs_out,
                *bgn_microsecs_out, *bgn_cycle_out,
                *end_orbit_out, *end_secs_out,
                *end_microsecs_out, *end_cycle_out,
                ierr[XV_NUM_ERR_DELTA], status;
       double   entry_offset, exit_offset;

      status = xv_time_segments_delta(
                        &orbit_id,
                        &orbit_type,
                        &entry_offset, &exit_offset,
                        &number_segments,
                        bgn_orbit, bgn_secs,
                        bgn_microsecs, bgn_cycle,
                        end_orbit, end_secs,
                        end_microsecs, end_cycle,
                        &num_segments_out,
                        &bgn_orbit_out, &bgn_secs_out,
                        &bgn_microsecs_out, &bgn_cycle_out,
                        &end_orbit_out, &end_secs_out,
                        &end_microsecs_out, &end_cycle_out,
                        ierr);

      /* Or, using the run_id */
      long run_id;
```

```
        status = xv_time_segments_delta_run(
                                 &run_id,
                                 &orbit_type,
                                 &entry_offset, &exit_offset,
                                 &number_segments,
                                 bgn_orbit, bgn_secs,
                                 bgn_microsecs, bgn_cycle,
                                 end_orbit, end_secs,
                                 end_microsecs, end_cycle,
                                 &num_segments_out,
                                 &bgn_orbit_out, &bgn_secs_out,
                                 &bgn_microsecs_out, &bgn_cycle_out,
                                 &end_orbit_out, &end_secs_out,
                                 &end_microsecs_out, &end_cycle_out,
                                 ierr);

}
```

7.25.3 Input parameters xv_time_segments_delta

Table 78: Input parameters of xv_time_segments_delta

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
entry_offset	double		Number of seconds to add/ subtract at the beginning of every segments. If entry_offset > 0, the entry_offset is added at the beginning of the segments making them shorter.	seconds	-
exit_offset	double		Number of seconds to add/ subtract at the end of every segments. If exit_offset > 0 the exit_offset is added at the end of the segments making them longer.	seconds	-
num_segments_in	long	-	Number of segments in the input list.	-	>0
bgn_orbit	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle	long*	all	Array of cycle numbers for the beginning of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL
end_orbit	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle	long*	all	Array of cycle numbers for the end of the segments. When using absolute orbits, a NULL pointer can be used.	-	>0 or NULL

7.25.4 Output parameters xv_time_segments_delta

Table 79: Output parameters of xv_time_segments_delta

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_del ta	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
num_segments_out	long	-	Number of segments in the output list.	-	>0
bgn_orbit_out	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs_out	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs_out	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle_out	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit_out	long*	all	Array of orbit numbers for the end of the segments	-	>0
end_secs_out	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs_out	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle_out	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
ierr[10]	long		Error status flags		

Memory Management: Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_time_segments_delta** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.25.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_delta** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_delta** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 80: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory	Computation not performed	XV_CFI_TIME_SEGMENTS_ DELTA_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_REL_TO_ABS_ERR	1
ERR	Error getting relative orbit vector from absolute orbits	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_ABS_TO_REL_ERR	2
ERR	Error transforming from orbits to processing times.	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_ORBIT_TO_TIME_E RR	3
ERR	Error transforming from processing times to orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_TIME_TO_ORBIT_E RR	4
ERR	Error modifying time segment duration	Computation not performed	XV_CFI_TIME_SEGMENTS_ DELTA_TIME_ADD_ERR	5
ERR	Error sorting input list	Computation not performed	XV_CFI_TIME_SEGMENTS_ DELTA_SORT_ERR	6
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_GEO_SAT_ERR	7
WARN	Deprecated function. Use xv_timesegments_compute_delta instead	Computation performed	XV_CFI_TIME_SEGMENTS_DELTA_DEPRECATED_WARN	8

7.26 xv_timesegments_compute_delta

7.26.1 Overview

An orbital segment is a time interval along the orbit, defined by start and stop times expressed as an orbit number and the seconds elapsed since the ascending node crossing.

The **xv_timesegments_compute_delta** function makes all the segments within a list, longer or shorter. After increasing/decreasing the longitude of the segments, these are sorted and merged to avoid possible overlapping. Therefore, at the end the list is sorted and without overlapped segments.

The time intervals (xv_time_interval) used by **xv_timesegments_compute_and** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times.

The **xv_timesegments_compute_delta** requires access to the following files to produce its results:

- the Orbit Scenario File: only if the orbits are expressed in relative numbers.

7.26.2 Calling sequence xv_timesegments_compute_delta

For C programs, the call to **xv_timesegments_compute_delta** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xo_orbit_id orbit_id = {NULL};
       xv_visibility_interval_list seg_in;
       xv_visibility_interval_list seg_out;
       long ierr[XV_NUM_ERR_COMPUTE_DELTA];
       long status;
       double entry_offset, exit_offset;

       status = xv_timesegments_compute_delta(
                        &orbit_id, &entry_offset, &exit_offset,
                        &seg_in,
                        &seg_out, ierr);


       /* Or, using the run_id */
       long run_id;

       status = xv_timesegments_compute_delta(
                        &run_id, &entry_offset, &exit_offset,
                        &seg_in,
                        &seg_out, ierr);

}
```

7.26.3 Input parameters xv_timesegments_compute_delta

Table 81: Input parameters of xv_timesegments_compute_delta

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit _id*	-	Structure that contains the orbit data	-	-
entry_offset	double		Number of seconds to add/ subtract at the beginning of every segments. If entry_offset > 0, the entry_offset is added at the beginning of the segments making them shorter.	seconds	-
exit_offset	double		Number of seconds to add/ subtract at the end of every segments. If exit_offset > 0 the exit_offset is added at the end of the segments making them longer.	seconds	-
seg_in	xv_visibilit y_interval_ list	-	Input list of segments	-	-

7.26.4 Output parameters xv_timesegments_compute_delta

Table 82: Output parameters of xv_timesegments_compute_delta

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_del ta	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
seg_out	xv_visibil ity_interv al_list	-	Output list of segments	-	-
ierr[]	long		Error status flags		

Memory Management: Note that the memory for the output visibility segments arrays is allocated within the **xv_timesegments_compute_delta** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.26.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_delta** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_delta** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 83: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_REL_TO_ABS_ERR	1
ERR	Error getting relative orbit vector from absolute orbits	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_ABS_TO_REL_ERR	2
ERR	Error transforming from orbits to processing times.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_ORBIT_TO_TIME_E RR	3
ERR	Error transforming from processing times to orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_TIME_TO_ORBIT_E RR	4
ERR	Error modifying time segment duration	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_TIME_ADD_ERR	5
ERR	Error sorting input list	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_SORT_ERR	6
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_DELTA_GEO_SAT_ERR	7
ERR	Wrong input time type	Computation not performed	XV_TIMESEGMENTS_COMPUTE_DELTA_TIME_TYPE_ERR	8
ERR	Error computing UTC time for orbit	Computation not performed	XV_TIMESEGMENTS_COMPUTE_DELTA_GET_UTC_T	9

			IME_ERR	

7.27 xv_time_segments_mapping

7.27.1 Overview

Note: this function is deprecated. Use **xv_timesegments_compute_mapping** instead.

The function **xv_time_segments_mapping** returns groups of visibility segments of a zone within an orbit range introduced by the user. These groups, or mappings, contain a minimum number of time segments needed to cover the zone completely, and fulfil the following conditions:

- Each mapping only contains ascending or descending segments.
- The segments are ordered by the track number.
- Mappings with one segment will be returned if it covers completely the zone.
- A mapping is searched for each track with segments that only contains left/right coverage in the case of ascending/descending segments, and finishes with a track that only contains right/left coverage.
- Incomplete mappings are not returned. This could happen if the number of orbits is insufficient to cover the zone.

Note that different mappings could contain a subset of segments in common. For example in Figure 30 there are two possible different mappings:

- mapping 1: orbits 1, 2, 3, 4.
- mapping 2: orbits 502, 2, 3, 4.

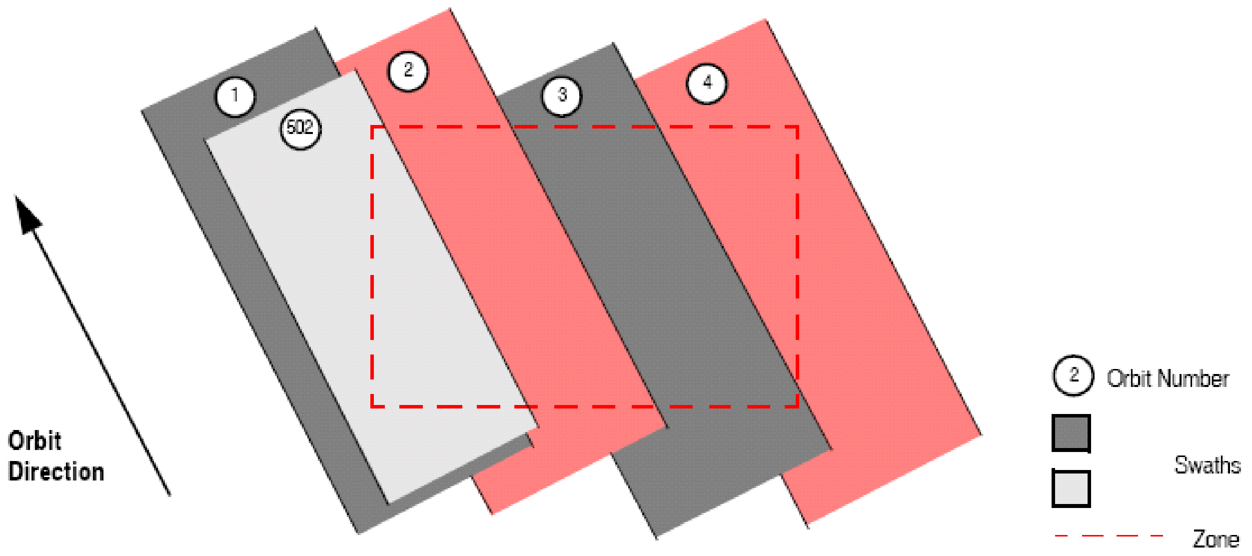


Figure 30: Different mappings with common segments

The time intervals used by **xv_time_segments_mapping** can be expressed in absolute or relative orbit numbers. This is valid for both:

- input parameter: first and last orbit to be considered. In case of using relative orbits, the corresponding cycle numbers should be used, otherwise, the cycle number will be a dummy parameter.

- output parameter: time segments with time expressed as {absolute orbit number (or relative orbit and cycle number), number of seconds since ANX, number of microseconds}

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits.

The **xv_time_segments_mapping** requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
- the Instrument Swath File, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit. The Swath data can be provided by:
  - A swath template file produced off-line by the EO_VISIBILITY library (**xv_gen_swath** function).
  - A swath definition file, describing the swath geometry. In this case the **xv_time_segments_mapping** generates the swath points for a number of orbits given by the user.
- Zone Database File: just in case of using a zone from the data base.

7.27.2 Calling sequence xv_time_segments_mapping

For C programs, the call to **xv_time_segments_mapping** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xo_orbit_id   orbit_id = {NULL};
       long     swath_flag, orbit_type,
                start_orbit, start_cycle,
                stop_orbit, stop_cycle,
                zone_num, projection;
                num_mappings, *num_segments,
                *orbit_direction,
                **bgn_orbit, **bgn_secs,
                **bgn_microsec, **bgn_cycle,
                **end_orbit, **end_secs,
                **end_microsec, **end_cycle,
                **coverage,
                ierr[XV_NUM_ERR_MAPPING], status;

       double   zone_diam, *zone_long, *zone_lat;

       char     *swath_file,
                zone_id[9], *zone_db_file;

       status = xv_time_segments_mapping(
                     &orbit_id, &orbit_type,
                     &start_orbit, &start_cycle,
                     &stop_orbit, &stop_cycle,
                     &swath_flag, swath_file,
                     &zone_num, zone_id, zone_db_file,
                     &projection, &zone_diam, zone_long, zone_lat,
                     &num_mappings, &num_segments,
                     &orbit_direction,
                     &bgn_orbit, &bgn_secs, &bgn_microsec, &bgn_cycle,
                     &end_orbit, &end_secs, &end_microsec, &end_cycle,
                     &coverage, ierr);


       /* Or, using the run_id */
       long run_id;
```

```
        status = xv_time_segments_mapping_run(
                      &run_id, &orbit_type,
                      &start_orbit, &start_cycle,
                      &stop_orbit, &stop_cycle,
                      &swath_flag, swath_file,
                      &zone_num, zone_id, zone_db_file,
                      &projection, &zone_diam, zone_long, zone_lat,
                      &num_mappings,           &num_segments,
                      &orbit_direction,
                      &bgn_orbit, &bgn_secs, &bgn_microsec, &bgn_cycle,
                      &end_orbit, &end_secs, &end_microsec, &end_cycle,
                      &coverage, ierr);
    }
```

7.27.3 Input parameters xv_time_segments_mapping

Table 84: Input parameters of xv_time_segments_mapping

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete (see Table 3)
start_orbit	long	-	First orbit, segment filter Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: • Absolute orbit is set to zero. • Relative orbit and cycle number set to zero.	absolute or relative orbit number	= 0 or: • absolute orbits ≥start_osf • relative orbits ≤ repeat cycle
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. The final orbit range defined by the start_orbit (start_cycle) and the stop_orbit (stop_cycle) should not exceed one cycle. Otherwise within one mapping there will appear all the orbits that are equal but that belong to different cycles. When: • stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS) • stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL) the stop_orbit will be set to the minimum value between: • the last orbit within the orbital change of the start_orbit. • start_orbit+cycle_length-1 (i.e.	absolute or relative orbit number	= 0 or: • absolute orbits ≥ start_osf • relative orbits ≤repeat cycle

			the input orbit range will be a complete cycle)		
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or ≥ first cycle in osf
swath_flag	long*	-	Define the use of the swath file: • 0 = (XV_STF) if the swath file is a swath template file. • > 0 if the swath files is a swath definition file. In this case the swath points are generated for every "swath_flag" orbits	-	XV_STF = 0 XV_SDF = 1 > 0
swath_file	char *	-	File name of the swath-file for the appropriate instrument mode		
zone_num	long		Number of vertices of the zone provided in zone_long, zone_lat: = 0 no vertices provided, use zone_id / zone_db_file = 1 Point / Circular zone, = 2 Line zone > 2 Polygon zone		≥ 0
zone_id[9]	char		Identification of the zone, as defined in zone_db_file. This parameter is used ONLY IF zone_num = 0		EXACTLY 8 characters
zone_db_file	char *		File name of the zone-database- file. This file is used ONLY IF zone_num = 0		
projection	long		projection used to define polygon sides as straight lines: = 0 Read projection from Zones DB (rectangular projection is used by default if the DB does not contain a projection) = 1 Azimuthal gnomonic = 2 Rectangular lat/long		
zone_diam	double		Zone diameter for circular zones, dummy for other zones If diameter equals 0.0 then zone is Point Zone	m	≥ 0.0
zone_long	double*	all	zone_long[i-1] Geocentric longitude of		

			- circle centre, for circ. zone, i =1 - point, for point zone, i = 1 - line-end, for line zone, i = 1 or 2 - vertices, for polygon zone, i = 1... zone_num		
zone_lat	double*	all	zone_lat[i-1] Geodetic latitude of - circle centre, for circ. zone, i =1 - point, for point zone, i = 1 - line-end, for line zone, i = 1 or 2 - vertices, for polygon zone, i = 1... zone_num		

7.27.4 Output parameters xv_time_segments_mapping

Table 85: Output parameters of xv_time_segments_mapping

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_mapping	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
num_mappings	long		Number of output mappings		≥ 0
num_segments	long*	all	num_segments[n] = number of segments for the n-th mapping. n=0... (num_mappings-1)	-	> 0
orbit_direction	long*	all	Direction of the segments of a mapping.	-	Complete (see Table 3: segment direction)
bgn_orbit	long**	all	Array of orbit numbers for the beginning of the segments	-	>0
bgn_secs	long**	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
bgn_microsecs	long**	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
bgn_cycle	long**	all	Array of cycle numbers for the beginning of the segments.	-	>0
end_orbit	long**	all	Array of orbit numbers for the end of the segments	-	>0
end_secs	long**	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
end_microsecs	long**	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
end_cycle	long**	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
coverage	long **	all	coverage of the output segments.	-	complete see Table 3
ierr	long*		Error status flags		

Note 1: The output visibility segments and the coverage are returned as a two-dimensional table where the first index indicates the number of the mapping, and the second one is the number of the segment within the mapping.

Note 2(Memory Management): Note that the output visibility segments arrays are pointers to integers instead of static arrays. The memory for these dynamic arrays is allocated within the **xv_time_segments_mapping** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.27.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_time_segments_mapping** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_time_segments_mapping** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 86: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error, wrong orbit Id.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_ORBIT_STATUS_ERR	0
ERR	Error getting absolute orbit from relative orbit.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_REL_TO_ABS_ERR	1
ERR	Error getting relative orbit vector from absolute orbits	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_REF_LATITUDE_ERR	2
ERR	Error computing swath width.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_WIDTH_ERR	3
ERR	Error calling zone_vis_time function	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_ZONEVISTIME_ERR	4
ERR	Error loading orbit scenario file.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_LOAD_OSF_ERR	5
ERR	Start orbit is less than first orbit in OSF	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_WRONG_START_ORB_ER R	6
ERR	Error, orbits changes found within the input orbit range	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_WRONG_STOP_ORB_ERR	7
ERR	Error allocating memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_MEM_ERR	8
ERR	Error sorting segments.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SORT_ERR	9

ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_ABS_TO_REL_ERR	10
ERR	Error checking extremes of the orbit range.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_CHECK_EXTREMES_ERR	11
ERR	Error calling xv_swath_pos function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_POS_ERR	12
ERR	Error loading swath template file: %s	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_READ_ERR	13
ERR	Swath file is not a line swath.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_INCORRECT_SWATH_ERR	14
WARN	Cannot check segments for start and stop orbits. Incomplete mappings could be generated.	Previous orbit to input start orbit and/or next orbit to the input stop orbit are not in the same orbital change that the input orbit range. It can not be checked whether there are segments missing at the extremes of the orbit range. Computation performed.	XV_CFI_TIME_SEGMENTS_MAPPING_NO_CHECK_PERFORMED_WARN	15
WARN	Incomplete ascending mapping. %ld more track(s) needed to complete the mapping.	Computation performed.	XV_CFI_TIME_SEGMENTS_MAPPING_ASC_INCOMPLETE_MAPPING_WARN	16
WARN	Incomplete descending mapping. %ld more track(s) needed to complete the mapping.	Computation performed.	XV_CFI_TIME_SEGMENTS_MAPPING_DESC_INCOMPLETE_MAPPING_WARN	17
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_GEO_SAT_ERR	18
WARN	Deprecated function. Use xv_timesegments_compute_mapping instead	Computation performed	XV_CFI_TIME_SEGMENTS_MAPPING_DEPRECATED_WARN	19

7.28 xv_timesegments_compute_mapping

7.28.1 Overview

The function **xv_timesegments_compute_mapping** returns groups of visibility segments of a zone within an orbit range introduced by the user. These groups, or mappings, contain a minimum number of time segments needed to cover the zone completely, and fulfil the following conditions:

- Each mapping only contains ascending or descending segments.
- The segments are ordered by the track number.
- Mappings with one segment will be returned if it covers completely the zone.
- A mapping is searched for each track with segments that only contains left/right coverage in the case of ascending/descending segments, and finishes with a track that only contains right/left coverage.
- Incomplete mappings are not returned. This could happen if the number of orbits is insufficient to cover the zone.

Note that different mappings could contain a subset of segments in common. For example in Figure 31 there are two possible different mappings:

- mapping 1: orbits 1, 2, 3, 4.
- mapping 2: orbits 502, 2, 3, 4.

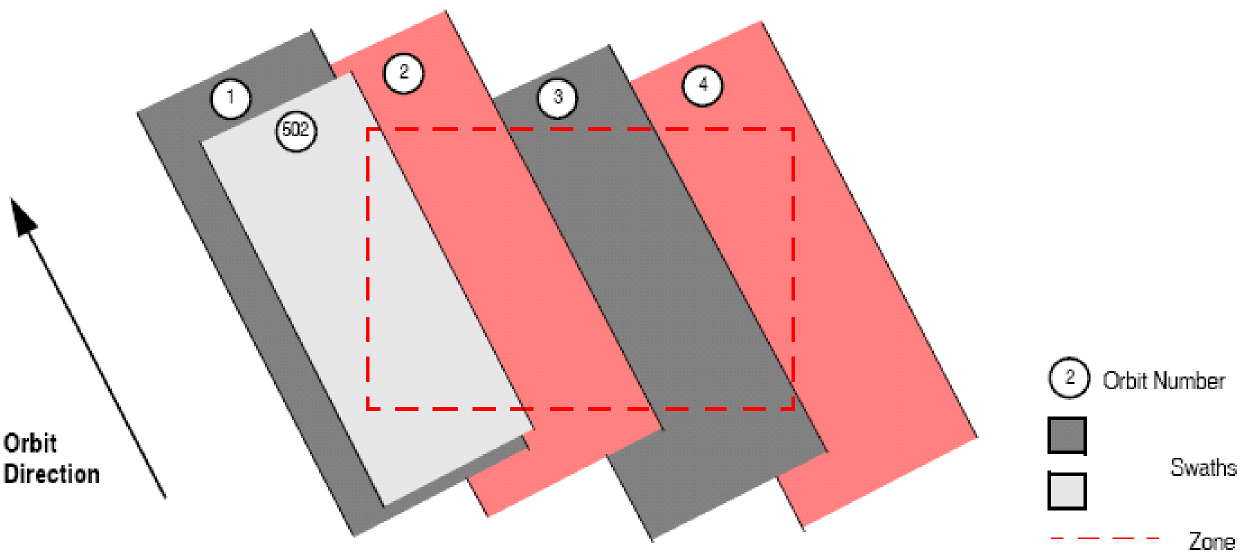


Figure 31: Different mappings with common segments

The time intervals (xv_time_interval) used by **xv_timesegments_compute_and** can be expressed as UTC times or orbit times (orbit plus seconds and microseconds since ascending node). This intervals express the start time/orbit and last time/orbit for the computations.

In case the time intervals are expressed as orbits, they can be expressed as absolute orbit numbers or in relative orbit and cycle numbers.

The orbit representation (absolute or relative) for the output segments will be the same as in the input orbits. The output segments will contain UTC times and orbit times.

The **xv_timesegments_compute_mapping** requires access to several data structures and files to produce its results:

- the orbit_id (xo_orbit_id) providing the orbital data. The orbit_id can be initialized with the following data or files (see [ORBIT_SUM]):
  - data for an orbital change
  - Orbit scenario files
  - Predicted orbit files
  - Orbit Event Files (**Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**)
  - Restituted orbit files
  - DORIS Preliminary orbit files
  - DORIS Navigator files
- The swath_id (xv_swath_id, initialized using xv_swath_id_init -section 7.32-), which provides the Instrument Swath data, excluding inertial swath files, describing the area seen by the relevant instrument all along the current orbit.
- Zone data (see section 7.3.7.5 for details)..

7.28.2 Calling sequence xv_timesegments_compute_mapping

For C programs, the call to **xv_timesegments_compute_mapping** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xo_orbit_id orbit_id = {NULL};
      xv_swath_id swath_id = {NULL};
       xv_zone_info_list zone_info_list;
       xv_time_interval search_interval;
       long num_mappings;
       long *orbit_direction = NULL;
       xv_zonevisibility_interval_list *seg_out = NULL;
       long ierr[XV_NUM_ERR_COMPUTE_MAPPING];

       status = xv_timesegments_compute_mapping(&orbit_id,
                               &swath_id, &zone_info_list,
                               &search_interval,
                               /*outpus*/
                               &num_mappings,
                               &orbit_direction, &seg_out,
                               ierr);


       /* Or, using the run_id */
       long run_id;

       status = xv_timesegments_compute_mapping(&run_id,
                               &swath_id, &zone_info_list,
                               &search_interval,
                               /*outpus*/
                               &num_mappings,
                               &orbit_direction, &seg_out,
                               ierr);
}
```

7.28.3 Input parameters xv_timesegments_compute_mapping

Table 87: Input parameters of xv_timesegments_compute_mapping

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
swath_id	xv_swath_id*	-	Swath id.	-	-
zone_info_list	xv_zone_info_list*	-	List of zones where the visibility is going to be computed.	-	-
search_interval	xv_time_interval*	-	Interval where the computations are performed	-	-

7.28.4 Output parameters xv_timesegments_compute_mapping

Table 88: Output parameters of xv_timesegments_compute_mapping

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_time_segments_mapping	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
num_mappings	long		Number of output mappings		≥ 0
orbit_direction	long*	all	Direction of the segments of a mapping.	-	Complete (see Table 3: segment direction)
seg_out	xv_visibility_interval_list*	-	Array with output list of segments. Every position corresponds to one of the mappings.	-	-
ierr	long*		Error status flags		

Note 1: The output visibility segments and the coverage are returned as a two-dimensional table where the first index indicates the number of the mapping, and the second one is the number of the segment within the mapping.

Note 2(Memory Management): Note that the memory for the output visibility segments arrays is allocated within the **xv_timesegments_compute_mapping** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.

7.28.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_timesegments_compute_mapping** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_timesegments_compute_mapping** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 89: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error, wrong orbit Id.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_ORB IT_STATUS_ERR	0
ERR	Error getting absolute orbit from relative orbit.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_REL_TO_ABS_ERR	1
ERR	Error getting relative orbit vector from absolute orbits	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_REF_LATITUDE_ERR	2
ERR	Error computing swath width.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_SWA TH_WIDTH_ERR	3
ERR	Error calling zone_vis_time function	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_ZON EVISTIME_ERR	4
ERR	Error loading orbit scenario file.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_LOAD_OSF_ERR	5
ERR	Start orbit is less than first orbit in OSF	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_WRONG_START_ORB_ER R	6
ERR	Error, orbits changes found within the input orbit range	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_WRONG_STOP_ORB_ERR	7
ERR	Error allocating memory.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_MEM_ERR	8
ERR	Error sorting segments.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_SO	9

			R T_ERR	
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_ABS_TO_REL_ERR	10
ERR	Error checking extremes of the orbit range.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_CHE CK_EXTREMES_ERR	11
ERR	Error calling xv_swath_pos function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_POS_ERR	12
ERR	Error loading swath template file: %s	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_READ_ERR	13
ERR	Swath file is not a line swath.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_INCORRECT_SWATH_ERR	14
WARN	Cannot check segments for start and stop orbits. Incomplete mappings could be generated.	Previous orbit to input start orbit and/or next orbit to the input stop orbit are not in the same orbital change that the input orbit range. It can not be checked whether there are segments missing at the extremes of the orbit range. Computation performed.	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_NO_CHECK_PERFORMED_WARN	15
WARN	Incomplete ascending mapping. %ld more track(s) needed to complete the mapping.	Computation performed.	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_ASC_INCOMPLETE_MAPPING_WARN	16
WARN	Incomplete descending mapping. %ld more track(s) needed to complete the mapping.	Computation performed.	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_DESC_INCOMPLETE_MAPPING_WARN	17
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_GEO_SAT_ERR	18
ERR	Swath id not initialized	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_SWATH_INIT_ERR	19
ERR	Error generating swath template file	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_GEN_SWATH_ERR	20
ERR	Wrong input time type	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_TIME_TYPE_ERR	21

ERR	Error computing time to orbit parameters	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_TIME_TO_ORBIT_ERR	22
ERR	Error computing UTC time for orbit	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_GET_UTC_TIME_ERR	23
ERR	Input file is not a swath file	Computation not performed	XV_CFI_TIMESEGMENTS_COMPUTE_MAPPING_DETECT_SWATH_TYPE_ERR	24

7.29 xv_gen_swath

7.29.1 Overview

The **xv_gen_swath** function generates for the different instrument modes the corresponding instrument swath template file. These template files define the swaths to be used in the segment calculation routines of EO_VISIBILITY.

The selection of the algorithm to compute the swath points depends on the parameters of the corresponding swath definition found in the instrument swath definition file. The swath point type (geodetic or inertial) and the algorithm to be used is deduced from the geometry and other instrument dependent parameters (see Table 90). There is an example of a swath definition file in the Appendix A.

The instrument swath template file, consists of a header which contains the altitude range of the swath. The data block contains *n* locations of the swath (between 50 and 6000, typically 1200) equally spread in time along one orbit. Every swath location contains a list of *m* points of the instantaneous swath (*m*>=1). For a description of the swath configuration see section 7.1.2 and Figure 8.

For Earth-fixed swaths, the location is given in longitude and latitude, in degrees, for the orbit with a longitude of ascending node of 0.0 degrees. For Inertial swaths, the location is the direction in inertial space (True of Date) in Right Ascension and Declination, in degrees, for the orbit with a Right Ascension of Ascending Node of 0.0 degrees.

The instrument swath template files are only dependent on:

- The instrument swath definition file
- The requested orbit number
- The orbit definition (orbit_id).

Table 90: Swath geometry definition (algorithm)

Geometry (XD_Swath_geom_enum)	Algorithm description	Swath point type (XD_Swath_point_type_enum)
Pointing_Geometry (azimuth, elevation, altitude)	Swath point computed with xp_target_inter with that azimuth, elevation and altitude	Geodetic
Distance_Geometry (azimuth, elevation, altitude, distance)	Swath point computed with xp_target_ground_range with that azimuth, elevation, altitude and distance	Geodetic
Limb_Geometry (azimuth and altitude)	Swath point computed with xp_target_altitude with that azimuth and altitude	Geodetic
Inertial_Geometry (azimuth and altitude)	Swath point computed with xp_target_altitude with that azimuth and altitude. The swath point is the RA and Declination of the target.	Inertial
Sub_Satellite_Geometry (no parameters)	Computation of the sub-satellite point	Geodetic
ASAR_Geometry (azimuth, elevation, altitude)	Specific algorithm for the three swath points for ASAR instrument in Envisat.	Geodetic

7.29.2 Calling interface

The calling interface of the **xv_gen_swath** CFI function is the following (input parameters are <u>underlined</u>):

```
#include <explorer_visibility.h>
{
      xo_orbit_id orbit_id = {NULL};
      xp_atmos_id atmos_id = {NULL};
      long requested_orbit,
           version_number;
      char *swath_definition_file;
      char swath_file[XD_MAX_STR], *dir_name, *file_class,
           *fh_system;
      long status, ierr[XV_ERR_VECTOR_MAX_LENGTH];

      status = xv_gen_swath (&orbit_id, &atmos_id,
                             &requested_orbit, swath_definition_file,
                             dir_name, swath_file,
                             file_class, &version_number, fh_system,
                             ierr);

      /* Or, using the run_id */
      long run_id;

      status = xv_gen_swath_run (&run_id,
                             &requested_orbit, swath_definition_file,
                             dir_name, swath_file,
                             file_class, &version_number, fh_system,
                             ierr);
}
```

7.29.3 Input parameters

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

Table 91: Input parameters of xv_gen_swath function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data.	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation. This parameter determines the atmospheric and raytracing model used for the STF generation. The refraction model in the SDF is not used.	-	-
requested_orbit	long*	-	Orbit for which the instrument swath template file will be calculated.	absolute orbit number	> 0
swath_definition_file	char*	-	File name of the instrument swath definition file	-	-
dir_name	char*	-	Directory where the resulting STF is written (if empty (i.e. ""), the current directory is used)	-	-
swath_file	char*	-	Name for output swath file. If empty (i.e. ""), the software will generate the name according to file name specification presented in [FORMATS], in this case the generated name is returned in this variable	-	-
file_class	char*	-	File class for output swath file	-	-
version_number	long*	-	Version number of output swath file	-	>= 1
fh_system	char*	-	System field of the output swath file fixed header	-	-

7.29.4 Output parameters

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

Table 92: Output parameters of xv_gen_swath function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
swath_file	char*	-	Name for output swath file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 91)	-	-
ierr[XV_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.29.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_gen_swath** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_gen_swath** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code** (see [GEN_SUM]).

Table 93: Error messages of xv_gen_swath function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error, wrong orbit Id.	Computation not performed	XV_CFI_GEN_SWATH_ORBIT_INIT_ERR	0
ERR	Wrong requested orbit	Computation not performed	XV_CFI_GEN_SWATH_REQUESTED_ORBIT_ERR	1
ERR	Could not get the creation date	Computation not performed	XV_CFI_GEN_SWATH_CURRENT_TIME_ERR	2
ERR	Error transforming time formats	Computation not performed	XV_CFI_GEN_SWATH_TIME_CONVERSION_ERR	3
ERR	Could not create the filename	Computation not performed	XV_CFI_GEN_SWATH_CREATE_FILENAME_ERR	4
ERR	Error reading swath definition file: %s	Computation not performed	XV_CFI_GEN_SWATH_SDF_READ_ERR	5
ERR	Error computing the swath points	Computation not performed	XV_CFI_GEN_SWATH_XV_ALGOR_ERR	6
ERR	Could not write the swath template file to disk	Computation not performed	XV_CFI_GEN_SWATH_WRITE_ERR	7
ERR	Wrong input file name. The file cannot be created	Computation not performed	XV_CFI_GEN_SWATH_WRONG_FILENAME_ERR	8
WARN	Could not find the input directory \"%s\". The current directory will be used instead	Computation performed	XV_CFI_GEN_SWATH_NO_DIR_WARN	9
ERR	Error allocating memory	Computation not performed	XV_CFI_GENSWATH_MEMORY_ERR	10
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_GENSWATH_GEO_SAT_ERR	11

7.29.6 Executable Program

The **gen_swath** executable program can be called from a Unix shell as:

gen_swath **-sat** satellite_name
-sdf swath_definition_file_name
-file orbit_file_name **-orbit** orbit_number
[-tle]
[**-dir** dir_name] (current directory by default)
[**-stf** swath_template_filename] (empty string by default)
[-precfile precise_configuration_file] (empty string by default)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-xp_v**]
[**-xv_v**]
[**-help**]
[**-show**]
{(-tai TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_reference_data file **-trid** time_reference
{(**-tm0** time 0 **-tm1** time 1) | (**-orb0** orbit 0 **-orb1** orbit 1) })**}**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory (For example, if **-stf** argument is not provided, instrument_swath_file_name_suffix is considered to be an empty string).
- Options between curly brackets and separated by a vertical bar are mutually exclusive (For example, that lines 3 and 4 are mutually exclusive).
- [-tle] this options must be provided if input file is a Two Line Elements file.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT Verbose mode.
- [**-xp_v**] option for EO_POINTING Verbose mode.
- [**-xv_v**] option for EO_VISIBILITY Verbose mode.
- [**-v**] option for Verbose mode for all libraries (default is Silent).
- [**-show**] displays the inputs of the function and the results.
- Possible values for *satellite_name*: ERS1, ERS2, ENVISAT, METOP1, METOP2, METOP3, CRYOSAT, ADM, GOCE, SMOS, TERRASAR, EARTHCARE, SWARM_A, SWARM_B,

SWARM_C, SENTINEL_1A, SENTINEL_1B, SENTINEL_1C, SENTINEL_2A, SENTINEL_2B, SENTINEL_2C, SENTINEL_3A, SENTINEL_3B, SENTINEL_3C, JASON_CSA, JASON_CSB, METOP_SG_A1, METOP_SG_A2, METOP_SG_A3, METOP_SG_B1, METOP_SG_B2, METOP_SG_B3, SENTINEL_5P, SEOSAT, GENERIC.

- Precise propagation is used if precfile is provided.
- **Important:** The atmospheric model used for the STF generation is taken from the "Refraction" parameters in the SDF. The allowed values for the Refraction model in the input SDF are NO_REF, STD_REF or PRED_REF. Note the user defined models are not allowed.

Example:

```
gen_swath -sat ENVISAT -orbit 2000 -osf ACCEPTANCE_OSF.N1
          -sdf SDF_MERIS.1200pts.N1 -xv_v
          -dir ./gen_swath
```

7.30 xv_gen_swath_no_file

7.30.1 Overview

The **xv_gen_swath_no_file** function generates for the different instrument modes the corresponding instrument swath template data.

The aim of this function is to provide another interface for the function **xv_gen_swath** in which the swath data is returned in a swath structure instead to be save to a file.

7.30.2 Calling interface

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

```
#include <explorer_visibility.h>
{
      xo_orbit_id orbit_id = {NULL};
      xp_atmos_id atmos_id = {NULL};
      long requested_orbit;
      xd_sdf_file *sdf;
      xd_stf_file *stf;

      long status, ierr[XV_ERR_VECTOR_MAX_LENGTH];
      status = xv_gen_swath_no_file (&orbit_id, &atmos_id,
                                     &requested_orbit,
                                     &sdf, &stf,
                                     ierr);

      /* Or, using the run_id */
      long run_id;

      status = xv_gen_swath_no_file_run (&run_id,
                                         &requested_orbit,
                                         &sdf, &stf,
                                         ierr);
}
```

7.30.3 Input parameters

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

Table 94: Input parameters of xv_gen_swath_no_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data.	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation. This parameters is needed only if the swath definition file requires atmosphere initialisation. This happens when the refraction model in the SDF is USER_REF or PRED_REF.	-	-
requested_orbit	long*	-	Orbit for which the instrument swath template file will be calculated.	absolute orbit number	> 0
sdf	xd_sdf_file	-	Swath definition file structure data. This structure is defined in [D_H_SUM] and can be got by reading a swath definition file with the CFI function **xd_read_sdf**.	-	-
file_class	char*	-	File class for output swath data	-	-
version_number	long*	-	Version number of output swath data	-	>= 1
fh_system	char*	-	System field of the output swath file fixed header data	-	-

7.30.4 Output parameters

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

Table 95: Output parameters of xv_gen_swath_no_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
stf	xd_stf_file	-	Swath Template structure defined in [D_H_SUM]	-	-

ierr[XV_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.30.5 Warnings and errors

The error and warning messages and codes for **xv_gen_swath_no_file** are the same than for **xv_gen_swath** (see Table 93) .

The error messages/codes can be returned by the CFI function **xv_get_msg/xv_get_code** after translating the returned status vector into the equivalent list of error messages/codes. The function identifier to be used in that functions is XV_GEN_SWATH_ID (from Table 2).

7.31 xv_gen_scf

7.31.1 Overview

The **xv_gen_scf** function generates a Swath Control file. This file contains a list of visibility segments together with some features linked to the segment that are used for the visualisation of the segment in the ESOV tool.

In order to generate the file, the same xo_orbit_id variable that was used for the generation of the visibility segments has to be provided. Moreover, this xo_orbit_id has to be implemented with one of the following functions:

- xo_orbit_init_def
- xo_orbit_init_file with an orbit scenario file (or an orbit event file used as an orbit scenario. **Note: Orbit Event File is deprecated, only supported for CRYOSAT mission**).

7.31.2 Calling interface

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

```
#include <explorer_visibility.h>
{
      xo_orbit_id orbit_id = {NULL};
      char instrument[XD_MAX_STR];
      long version_number;
      char *file_class,*fh_system;
      char dir_name[XD_MAX_STR], scf_filename[XD_MAX_STR];
      long status, ierr[XV_NUM_ERR_GEN_SCF];
      long number_segments;
      long *bgn_orbit, *bgn_second, *bgn_microsec;
      long *end_orbit, *end_second, *end_microsec;
      xd_scf_appear * appearance;
      status = xv_gen_scf (&orbit_id, instrument, &number_segments,
                           bgn_orbit, bgn_second, bgn_microsec,
                           end_orbit, end_second, end_microsec,
                           appearance,
                           dir_name, scf_filename,
                           file_class, &version_number, fh_system,
                           ierr);

      /* Or, using the run_id */
      long run_id;
      status = xv_gen_scf_run (&run_id, instrument, &number_segments,
                               bgn_orbit, bgn_second, bgn_microsec,
```

```
                                    end_orbit, end_second, end_microsec,
                                    appearance,
                                    dir_name, scf_filename,
                                    file_class, &version_number,fh_system,
                                    ierr);
    }
```

7.31.3 Input parameters

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

Table 96: Input parameters of xv_gen_scf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data.	-	-
instrument	char*	-	Instrument name	-	-
number_segments	long	-	Number of input segments	-	
bgn_orbit	long*	-	Array of absolute orbit numbers for the beginning of the segments	-	> 0
bgn_second	long*	-	Array of seconds elapsed since ANX for the beginning of the segments	-	> =0
bgn_microsec	long*	-	Array of microseconds within a second for the beginning of the segments	-	> =0
end_orbit	long*	-	Array of absolute orbit numbers for the end of the segments	-	> 0
end_second	long*	-	Array of seconds elapsed since ANX for the end of the segments	-	> =0
end_microsec	long*	-	Array of microseconds within a second for the end of the segments	-	> =0
appearance	xd_scf_appe ar	-	Array with the structures contain ing the appearance for every segment (see [D_H_SUM])	-	-
dir_name	char*	-	Directory where the resulting STF is written (if empty (i.e. ""), the current directory is used)	-	-
scf_filename	char*	-	Name for output swath file. If empty (i.e. ""), the software will generate the name according to file name specification presented	-	-

			in [FORMATS], in this case the generated name is returned in this variable		
file_class	char*	-	File class for output file	-	-
version_number	long*	-	Version number of output file	-	>= 0
fh_system	char*	-	System field of the output file fixed header	-	-

7.31.4 Output parameters

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

Table 97: Output parameters of xv_gen_scf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
scf_filename	char*	-	Name for output SCF. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 96)	-	-
ierr[XV_NUM_ERR_GEN_SCF]	long	all	Status vector	-	-

7.31.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_gen_scf** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_gen_scf** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code** (see [GEN_SUM]).

Table 98: Error messages of xv_gen_scf function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	No segments to write	Computation not performed	XV_CFI_GENSCF_NO_SEGMENTS_ERR	0
ERR	The orbit has not been initialised	Computation not performed	XV_CFI_GENSCF_ORBIT_INIT_ERR	1
ERR	Wrong orbit initialisation mode	Computation not performed	XV_CFI_GENSCF_ORBIT_INIT_MODE_ERR	2
ERR	Could not get the creation date	Computation not performed	XV_CFI_GENSCF_CURRENT_TIME_ERR	3
ERR	Could not get orbit number for the orbit = %ld	Computation not performed	XV_CFI_GENSCF_ORBIT_TO_TIME_CONVERSION_E RR	4
ERR	Error transforming time formats	Computation not performed	XV_CFI_GENSCF_TIME_CONVERSION_ERR	5
ERR	Could not create the filename	Computation not performed	XV_CFI_GENSCF_CREATE_FILENAME_ERR	6
ERR	Could not get orbital information for orbit %ld	Computation not performed	XV_CFI_GENSCF_GET_ORBIT_INFO_ERR	7
ERR	Wrong input file name. The file cannot be created	Computation not performed	XV_CFI_GENSCF_WRONG_FILENAME_ERR	8
ERR	Could not write the swath control file to disk	Computation not performed	XV_CFI_GENSCF_WRITE_ERR	9
WARN	Could not find the input directory \"%s\". The current directory will be used instead	Computation performed	XV_CFI_GENSCF_NO_DIR_WARN	10

7.32 xv_swath_id_init

7.32.1 Overview

The **xv_swath_id_init** function initializes the swath ID. It can be initialized with the following data:

- Swath definition data
- Swath template data
- Swath definition file
- Swath template file

Once initilized, the swath data is stored in internal structures pointed by swath id.

7.32.2 Calling sequence of xv_swath_id_init

For C programs, the call to **xv_swath_id_init** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
      xv_swath_info swath_info;
      xp_atmos_id atmos_id = {NULL};
      xv_swath_id swath_id = {NULL};
      long ierr[XV_NUM_ERR_SWATH_ID_INIT], status;

      status = xv_swath_id_init(&swath_info,
                                &atmos_id,
                                &swath_id,
                                ierr);

}
```

7.32.3 Input parameters xv_swath_id_init

Table 99: Input parameters of xv_swath_id_init

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
swath_info	xv_swath_info	-	Swath initialisation information	-	-
atmos_id	xp-atmos_id	-	Atmos ID	-	-

7.32.4 Output parameters xv_swath_id_init

Table 100: Output parameters of xv_swath_id_init

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_swath_id_init	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
swath_id	xv_swath_id	-	Swath ID.		
ierr	long[]		Error status flags		

7.32.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_swath_id_init** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_swath_id_init** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 101: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input initialization type	Computation not performed	XV_CFI_SWATH_ID_INIT_TYPE_ERR	0
ERR	Expected input data not found for input type	Computation not performed	XV_CFI_SWATH_ID_INIT_WRONG_INPUT_COMBINATION_ERR	1
ERR	Error: swath id is initialized	Computation not performed	XV_CFI_SWATH_ID_INIT_SWATH_INIT_STATUS_ERR	2
ERR	Error allocating memory	Computation not performed	XV_CFI_SWATH_ID_INIT_MEMORY_ERR	3
ERR	Error cloning swath definition structure	Computation not performed	XV_CFI_SWATH_ID_INIT_SDF_CLONE_ERR	4
ERR	Error cloning swath template structure	Computation not performed	XV_CFI_SWATH_ID_INIT_STF_CLONE_ERR	5
ERR	Error linking ids	Computation not performed	XV_CFI_SWATH_ID_INIT_LINK_ID_ERR	6

7.33 xv_swath_id_close

7.33.1 Overview

The **xv_swath_id_close** function frees the data allocated in the swath ID.

7.33.2 Calling sequence of xv_swath_id_close

For C programs, the call to **xv_swath_id_close** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
        xv_swath_id swath_id = {NULL};
        long ierr[XV_NUM_ERR_SWATH_ID_CLOSE], status;

        status = xv_swath_id_close(&swath_id, ierr);

}
```

7.33.3 Input parameters xv_swath_id_close

Table 102: Input parameters of xv_swath_id_close

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
swath_id	xv_swath_id	-	Swath ID.		

7.33.4 Output parameters xv_swath_id_close

Table 103: Output parameters of xv_swath_id_close

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_swath_id_close	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
ierr	long[]		Error status flags		

7.33.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_swath_id_close** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_swath_id_close** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 104: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Could not close the swath id	Computation not performed	XV_CFI_SWATH_ID_CLOSE_WRONG_ID_ERR	0

7.34 xv_swath_set_id_data

7.34.1 Overview

The **xv_swath_set_id_data** function sets the values of the swath ID internal structure according to inputs.

7.34.2 Calling sequence of xv_swath_set_id_data

For C programs, the call to **xv_swath_set_id_data** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xv_swath_id swath_id = {NULL};
       xv_swath_info swath_info;

       status = xv_swath_set_id_data(&swath_id, &swath_info);

}
```

7.34.3 Input parameters xv_swath_set_id_data

Table 105: Input parameters of xv_swath_set_id_data

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
swath_id	xv_swath_id	-	Swath ID.		
swath_info	xv_swath_info	-	Swath information		

7.34.4 Output parameters xv_swath_set_id_data

Table 106: Output parameters of xv_swath_set_id_data

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_swath_set_id_data	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		

7.34.5 Warnings and errors

This function does not return any errors.

7.35 xv_swath_get_id_data

7.35.1 Overview

The **xv_swath_get_id_data** function gets the values stored in the swath ID internal structure.

7.35.2 Calling sequence of xv_swath_get_id_data

For C programs, the call to **xv_swath_get_id_data** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
        xv_swath_id swath_id = {NULL};
        xv_swath_info swath_info;

        status = xv_swath_get_id_data(&swath_id, &swath_info);

}
```

7.35.3 Input parameters xv_swath_get_id_data

Table 107: Input parameters of xv_swath_get_id_data

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
swath_id	xv_swath_id	-	Swath ID.		

7.35.4 Output parameters xv_swath_get_id_data

Table 108: Output parameters of xv_swath_get_id_data

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_swath_get_id_data	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
swath_info	xv_swath_info	-	Swath information		

7.35.5 Warnings and errors

This function does not return any errors.

7.36 xv_zonevistime_coverage

7.36.1 Overview

The **xv_zonevistime_coverage** function computes the portion of the input zone that is covered by a swath during a set of input time visibility intervals (also called segments). The user has to initialize the input structure of type xv_zonevisibility_coverage_in (see Table 4) in order to provide the required information for the computation: zone, swath, attitude definition, orbit_id, list of visibility times. Note that the *visibility_interval_list* field is of type xv_zonevisibility_interval_list (see Table 4) that is the output of the xv_zonevistime_compute (section 7.36) function.

The following outputs are returned in the xv_zonevisibility_coverage_out struct (assuming that intervals are numbered from 0 to N-1 as stored in the array visibility_interval within visibility_interval_list, N is the num_rec field within visibility_interval_list):

1) The **area** of the zone in Km^2 (zone_area field);
2) The **total coverage** (total_coverage field): this is the percentage of zone covered by all intervals, i.e. 100 minus the percentage of zone not covered by any interval;
3) The **coverage per interval** (coverage_per_interval field): this is an array of size N that contains the coverage percentages computed considering only one interval. Item of the array with index i (0,1,2,......N-1) is the percentage of zone covered by the interval i (0,1,2,...N-1) only.
4) The **coverage per number of intervals (**coverage_by_N_intervals field): this is an array of size N that contains the percentages of zone covered by exactly 1,2,...,N intervals. Item with index i (0,1,2,...N-1) is the percentage of the zone covered by exactly i+1 (1,2,...N) intervals. This array can be used to evaluate how much segments are overlapping. Ideally, with segments not overlapping, only the first item of this array should be different from zero, and should be close to the total coverage. If other items of the array are different from zero, this means that portions of the zone are covered by more than one segment;
5) The **cumulative coverage** (cumulative_coverage field): this is an array of size N that contains the cumulative coverage percentage. Item with index i (0,1,2,...N-1) is the percentage of zone covered by intervals 0,1,2…i+1 considered together. This output considers the order in which time segments are provided. For example, if intervals are sorted by time, this output can be used to evaluate when a target coverage percentage is reached.

The computation is performed by means of a grid of points inside the zone. The portion of these points that are within the swath gives the percentage of area covered. The precision of the computation then depends on the distance between points. Two ways of computing the points can be provided by the user via the type_coverage field:

1) XV_COVERAGE_FIXED_DISTANCE: a fixed distance between points is used. The value of these distance is provided by the user (in kilometers);
2) XV_COVERAGE_PERCENTAGE_PRECISION: the required percentage precision is provided by the user. The function computes internally the distance needed to achieve the requested precision. The distance is computed in a way that the area of the zone computed with the grid created with such distance

is close to the expected one (computed by analytical methods) by a percentage equal or better than input percentage precision. That is, if *area_zone* is the zone area computed analytically and *area_grid* is the area of the zone computed with the grid, the accuracy percentage is defined as:

percen_accuracy = 100. * (1- abs((area_zone-area_grid) / area_zone))

The selected distance fulfills that *percen_accuracy* is greater or equal to input percentage precision. That means that the closer the input percentage precision to 100% the more accurate are the computations.

WARNING: The user has to be aware that requesting a small distance or a big value for precision increase the number of points to be analyzed and this has an impact on performances.

The following example is presented to explain how this function operates:

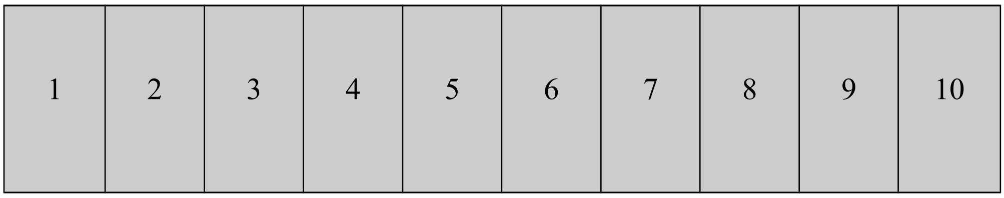


Fig. 1

Zone split into 10 tiles

A simplified representation of the input zone is given in Fig. 1: the zone is split into 10 tiles of same size.

Three time intervals are provided as input, they are identified with their indexes: 0,1,2

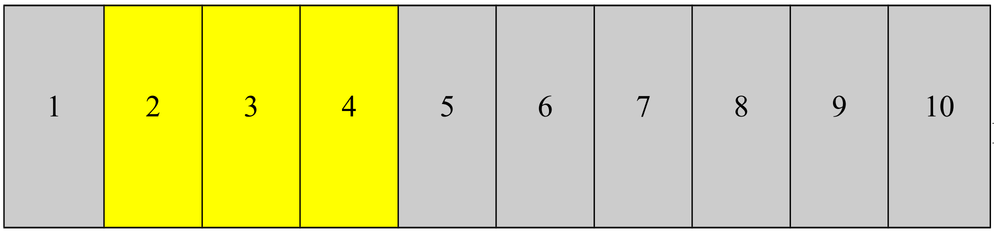


Fig. 2

Portion of Zone covered by segment 0 only

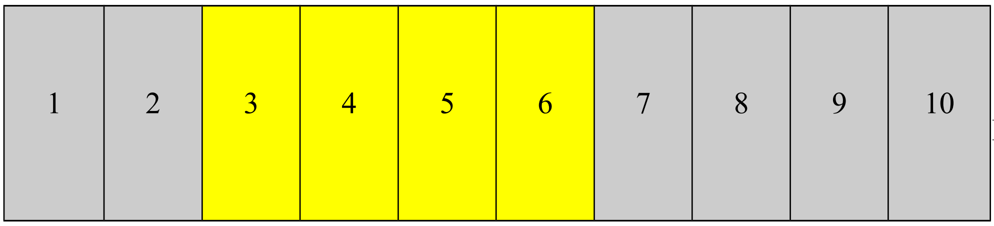


Fig. 3

Portion of Zone covered by segment 1 only

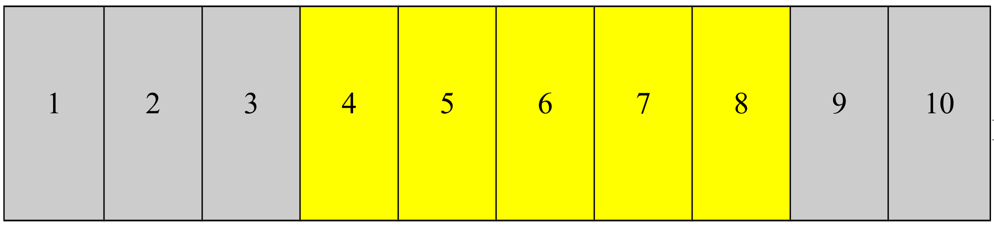


Fig. 4

Portion of Zone covered by segment 2 only

During time interval 0, tiles 2,3,4 are covered, see Fig. 2.

During time interval 1, tiles 3,4,5,6 are covered, see. Fig. 3

During time interval 2, tiles 4,5,6,7,8 are covered, see Fig. 4.

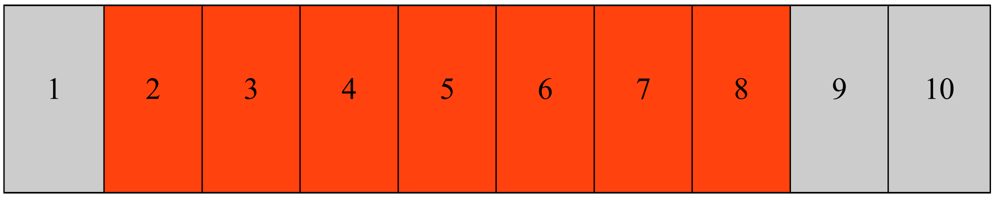


Fig. 5

Total coverage

Considering all three segments, 7 tiles of the zone (2 to 8) are covered, this corresponds to 70% of the whole zone (see Fig. 5). Therefore, if *cov_out* is the output variable of type xv_zonevisibility_coverage_out:

cov_out.total_coverage = 70

intervals 0,1,2 cover respectively 3,4,5 tiles corresponding to 30%, 40%, 50% of the whole zone. Therefore:

cov_out.coverage_per_interva[0] = 30

cov_out.coverage_per_interva[1] = 40

cov_out.coverage_per_interva[2] = 50

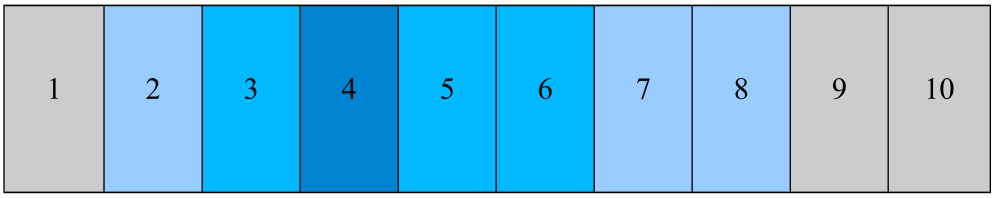


Fig. 6

Coverage by N segments

Fig. 6 details the computation of the coverage_by_N_intervals field. Tiles with same colors are covered by the same number of segments.

Tile 2 is covered only by interval 0. Tiles 7,8 are covered only by interval 4.

Therefore tiles 2,7,8 are covered by only one interval. This corresponds to 30% of the whole zone. Therefore:

cov_out.coverage_by_N_intervals[0] = 30

Similarly tiles 3,5,6 are covered by only 2 intervals and tile 4 is covered by all 3 segments:

Therefore:

cov_out.coverage_by_N_intervals[1] = 30

cov_out.coverage_by_N_intervals[2] = 10

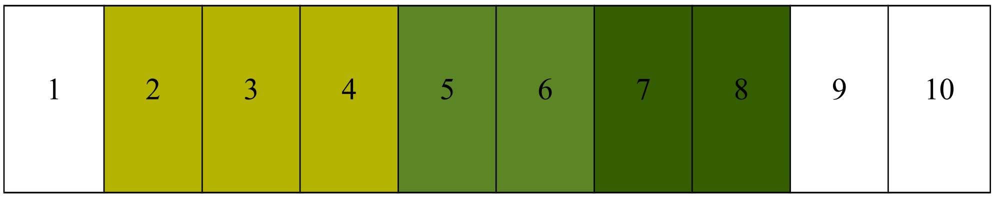


Fig. 7

Cumulative Coverage

Fig. 7 details the computation of the cumulative_coverage field. Tiles with same color are those tiles that are covered by a segment but not but the previous ones in the list.

Item 0 is the coverage of the first segment only, that is tiles 2,3,4, i.e. 30%.

Item 1 is the cumulative coverage of the first and second segments, that is tiles 2,3,4,5,6, that is 50%.

item 2 is the cumulative coverage of the first the second and third segment, that is tiles 2,3,4,5,6,7,8 that is 70%. Therefore:

cov_out.cumulative_coverage [0] = 30

cov_out.cumulative_coverage [1] = 50

cov_out.cumulative_coverage [2] = 70

7.36.2 Calling sequence of xv_zonevistime_coverage

For C programs, the call to **xv_zonevistime_coverage** is (input parameters are underlined):

```
#include"explorer_visibility.h"
{
       xv_zonevisibility_coverage_in  zone_cov_in;
       xv_zonevisibility_coverage_out zone_cov_out;
       long ierr[XV_NUM_ERR_ZONEVISTIME_COVERAGE], status;

       status = xv_zonevistime_coverage(&zone_cov_in,
                                  &zone_cov_out,
                                  ierr);

}
```

7.36.3 Input parameters xv_zonevistime_coverage

Table 109: Input parameters of xv_zonevistime_coverage

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
zone_cov_in	xv_zonevisibility_coverage_in	-	Inputs for zone coverage computations	-	-

7.36.4 Output parameters xv_zonevistime_coverage

Table 110: Output parameters of xv_zonevistime_coverage

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_zonevistime_coverage	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
zone_cov_out	xv_zonevistime_coverage_out	-	Zone coverage output.		
ierr	long[]		Error status flags		

Note: in the output struct xv_zonevistime_coverage_out, all the arrays are allocated dynamically and the user is responsible for freeing this allocated memory.

7.36.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_zonevistime_coverage** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_VISIBILITY software library **xv_get_msg**.

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, mainly on the results vector.

The table is completed by the error code and value. These error codes can be obtained translating the status vector returned by the **xv_zonevistime_coverage** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 111: Error messages and codes

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Name of zone id is empty.	Computation not performed	XV_CFI_ZONEVISTIME_COVERAGE_ZONE_ID_EMPTY_ERR	0
ERR	Error reading zone database.	Computation not performed	XV_CFI_ZONEVISTIME_COVERAGE_READ_ZONE_DB_FILE_ERR	1
ERR	Error cloning zone	Computation not performed	XV_CFI_ZONEVISTIME_COVERAGE_CLONE_ZONE_ERR	2
ERR	Error creating point database	Computation not performed	XV_CFI_ZONEVISTIME_COVERAGE_CREATE_DB_ERR	3
ERR	Error allocating memory	Computation not performed	XV_CFI_ZONEVISTIME_COVERAGE_MEMORY_ERR	4
ERR	Error in zonevistime computations	Computation not performed	XV_CFI_ZONEVISTIME_COVERAGE_ZVT_COMPUTE_ERR	5

8 RUNTIME PERFORMANCES

The library performance has been measured by dedicated test procedures run in 5 different platforms under the below specified machines:

OS ID	*Processor*	*OS*	*RAM*
LINUX64	Intel(R) Xeon(R) CPU E5-2470 0 @ 2.30GHz (16 cores)	GNU LINUX 2.6.24-16-generic (Ubuntu 8.04)	16 GB
LINUX32_LEGACY	Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz	GNU LINUX 2.6.24-16-generic (Ubuntu 8.04)	4 GB
LINUX64_LEGACY	Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz	GNU LINUX 2.6.24-16-generic (Ubuntu 8.04)	4 GB
MACIN64	Intel Core i7 4 cores @2,6 GHz	MAC OSX V10.9	16 GB
WINDOWS	Intel(R) Core(TM)2 i5-2450M CPU @ 2.50GHz 2.50GHz	Microsoft Windows 7	6 GB

The table below shows the time (in miliseconds - ms) each function takes to be run under each platform:

Function ID	**WINDOWS**	**LINUX64**	**LINUX64_LEGACY**	**LINUX32_LEGACY**	**MACIN64**
xv_swathpos_compute	0.050800	0.034000	0.062000	0.045000	0.019000
xv_zonevistime_compute * 50 orbits	36.230000	28.299999	46.400002	42.000000	34.900002
xv_zonevistime_coverage * Percentage precision = 75% (5 segments)	2641.000000	2310.000000	3770.000000	3330.000000	2990.000000
xv_timesegments_compute_not * 34 segments	0.033600	0.021600	0.026800	0.043300	0.013700
xv_timesegments_compute_or * 34 segments	0.038110	0.023300	0.031400	0.047100	0.016800
xv_timesegments_compute_and * 34 segments	0.056130	0.026200	0.032300	0.051300	0.019200
xv_timesegments_compute_sort * 34 segments	0.051000	0.030000	0.030000	0.040000	0.020000
xv_timesegments_compute_merge * 34 segments	0.048160	0.021900	0.027200	0.044100	0.013900
xv_timesegments_compute_delta * 34 segments	11.051000	7.740000	7.920000	11.570000	4.740000
xv_zonevistime_compute (6 zones) * 30 orbits 6 zones	44.900002	34.000000	55.000000	57.000000	41.000000
xv_gen_scf * 27 segments	4.234000	1.570000	2.070000	2.640000	1.980000
xv_station_compute * 10 orbits 7 stations	166.300003	117.000000	174.000000	185.000000	120.000000
xv_stationvistime_compute * 10 orbits	41.200001	33.000000	51.000000	52.000000	39.000000

xv_star_vis_time * 100 orbits	335.200012	316.000000	505.000000	531.000000	355.000000
xv_timesegments_compute_mapping * 50 orbits	156.399994	139.000000	235.000000	254.000000	179.000000
xv_orbit_extra	20.020000	11.400000	16.799999	17.299999	7.400000
xv_gen_swath	321.470001	165.399994	240.500000	282.399994	168.199997
xv_gen_swath_no_file	229.149994	95.199997	126.099998	142.199997	77.000000
xv_sc_vis_time * 10 orbits	902.700012	491.000000	656.000000	708.000000	305.000000

Note that when the value “0.000000” is defined for a function in a certain platform, it means that its running time is lower than 1 nano-second and so it can be considered as “0”.LIBRARY PRECAUTIONS

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

- When a message like
  <LIBRARY NAME> >>> ERROR in *xv_function*: Internal computation error # *n*
  or
  <LIBRARY NAME> >>> WARNING in *xv_function*: Internal computation warning # *n*
  appears, run the program in ***verbose*** mode for a complete description of warnings and errors and call for maintenance if necessary.

9 KNOWN PROBLEMS

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

Table 112: Known problems

CFI library	Problem	Work around solution
xv_gps_vis_time	Functions not available yet	-