Earth Observation Mission CFI Software

EO_VISIBILITY SOFTWARE USER MANUAL

Code:	EO-MA-DMS-GS-0006
Issue:	4.2
Date:	31/01/2011

	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.
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:	OpenOffice 2.0
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		

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

[GEN_SUM]	Earth Observation Mission CFI Software. General Software User Manual. EO-MA- DMS-GS-0002. Issue 4.2 31/01/2011.

3.2 Reference Documents

[MCD]	Earth Observation Mission CFI Software. Conventions Document. EO-MA-DMS-GS-0001. Issue 4.2 31/01/2011.
[MSC]	Earth Observation Mission CFI Software. Mission Specific Customizations Document. EO-MA- DMS-GS-0018. Issue 4.2 31/01/2011.
[F_H_SUM]	Earth Observation Mission CFI Software. EO_FILE_HANDLING Software User Manual. EO-MA-DMS-GS-0008. Issue 4.2 31/01/2011.
[D_H_SUM]	Earth Observation Mission CFI Software. EO_DATA_HANDLING Software User Manual. EO-MA-DMS-GS-007. Issue 4.2 31/01/2011.
[LIB_SUM]	Earth Observation Mission CFI Software. EO_LIB Software User Manual. EO-MA-DMS-GS-003. Issue 4.2 31/01/2011.
[ORBIT_SUM]	Earth Observation Mission CFI Software. EO_ORBIT Software User Manual. EO-MA-DMS-GS-0004. Issue 4.2 31/01/2011.
[POINT_SUM]	Earth Observation Mission CFI Software. EO_POINTING Software User Manual. EO-MA-DMS-GS-0005. Issue 4.2 31/01/2011.
[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_station_vis_time** and **xv_station_vis_time_no_file**: compute visibility time segments for a ground station
- **xv_drs_vis_time**: computes visibility time segments for a data relay satellite
- **xv_zone_vis_time** and **xv_zone_vis_time_no_file**: compute visibility time segments for an instrument swath in visibility of a zone.
- **xv_swath_pos**: computes location of a swath at a given time (additional routine to help refine the results of **xv_zone_vis_time**)
- **xv_star_vis_time**: computes visibility time segments for a star.
- xv_multizones_vis_time: computes the visibility segments of several zones and sort them to different criteria.
- **xv_multistations_vis_time**: computes the visibility segments of several ground stations and sort them according to different criteria.
- **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_time_segments_not:** returns the complement of 1 vector of time segments.
  - **xv_time_segments_and:** returns the intersection segments from 2 vectors of time segments.
  - **xv_time_segments_or:** returns the joined segments from 2 vectors of time segments
  - **xv_time_segments_delta:** add or subtract time durations at the beginning and end of each time segment in a vector.
  - **xv_time_segments_sort:** returns the vector of time segments sorted according to absolute or relative orbits.
  - **xv_time_segments_merge:** merges all the overlapped segments in a list.

- **xv_time_segments_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_station_vis_time, xv_zone_vis_time, xv_swath_pos**)
- Ground Stations Database File (**xv_station_vis_time**)
- (optionally) Zones Database File (**xv_zone_vis_time**)
- (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

4.2 Calling Sequence

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

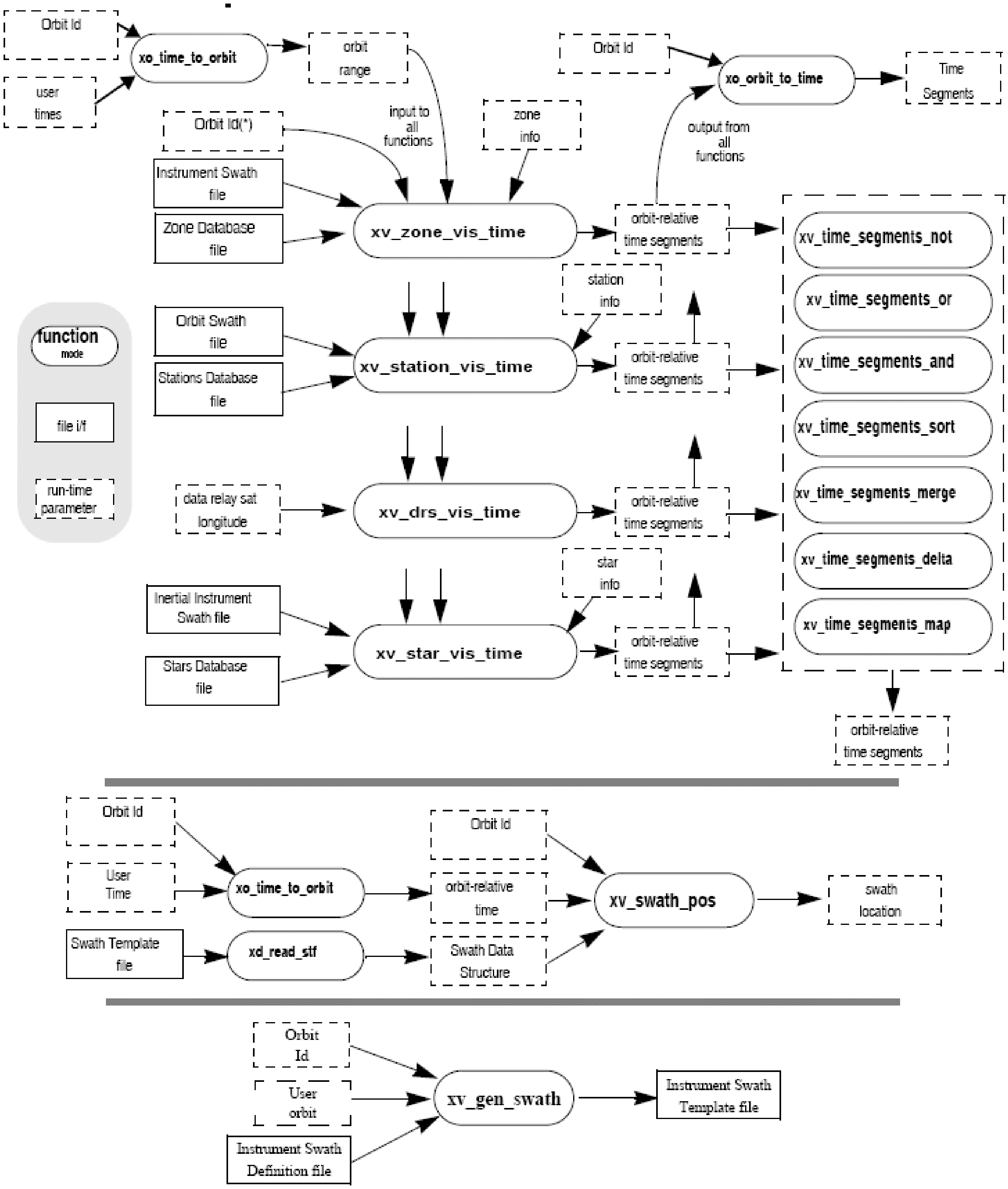

Figure 1: EO_VISIBILITY Data Flow

*orbit_id initialised with TLE or precise modes are not supported.

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

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

- `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])

It is needed to have properly installed in the system the following external libraries:

- `LIBXML2` (MIT license, see [GEN_SUM]).
- POSIX thread library: libpthread.so (pthread.lib for WINDOWS, with license LGPL)

To use the EO_VISIBILITY software library in a user application, that application must include in his source code :

- `explorer_visibility.h` (for a C application)

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

- SOLARIS/LINUX:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_visibility
-lexplorer_pointing -lexplorer_orbit -lexplorer_lib
-lexplorer_data_handling -lexplorer_file_handling
-lxml2 -lpthread
```

- Windows users:

```
/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
                         libxml2.lib pthread.lib
```

- MacOS:

```
-Icfi_include_dir -Lcfi_lib_dir -lexplorer_visibility
```

```
-lexplorer_pointing -lexplorer_orbit -lexplorer_lib
-lexplorer_data_handling -lexplorer_file_handling
 -framework libxml -framework libiconv -lpthread
```

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 1.

Table 1: 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_drs_vis_time	XV_DRS_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_swath xv_gen_swath_no_file	XV_GEN_SWATH_ID	15
xv_gen_scf	XV_GEN_SCF_ID	16
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 2: 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

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**.

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

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

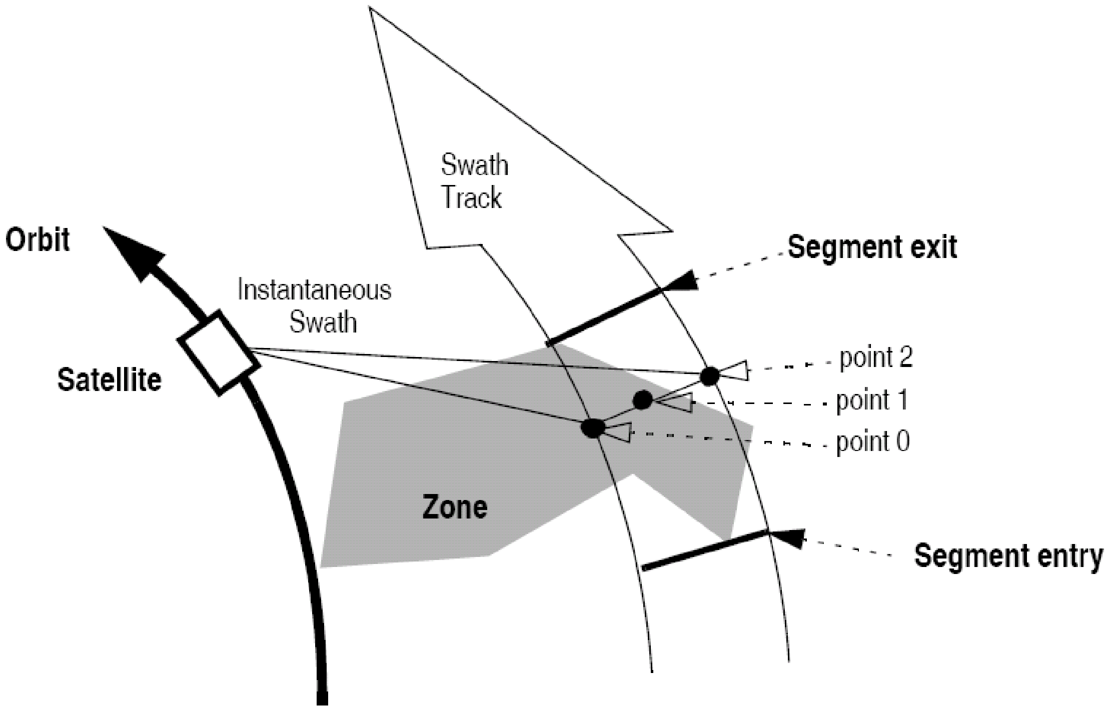

Figure 2: Segment Definition xv_zone_vis_time

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

- 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 46). 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.

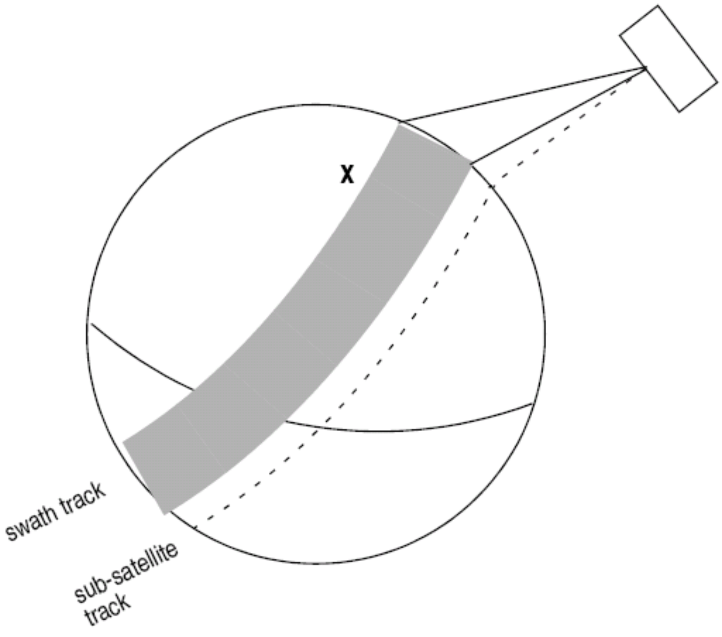

Figure 3: Earth-observing instrument: swath definition

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

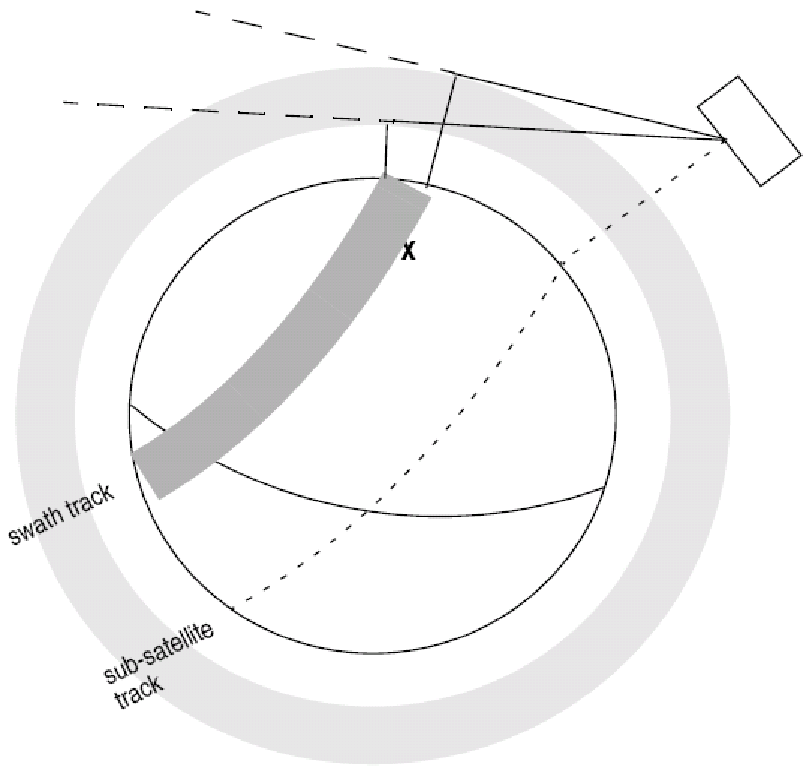

Figure 4: Limb-sounding instrument: swath definition (1)

.

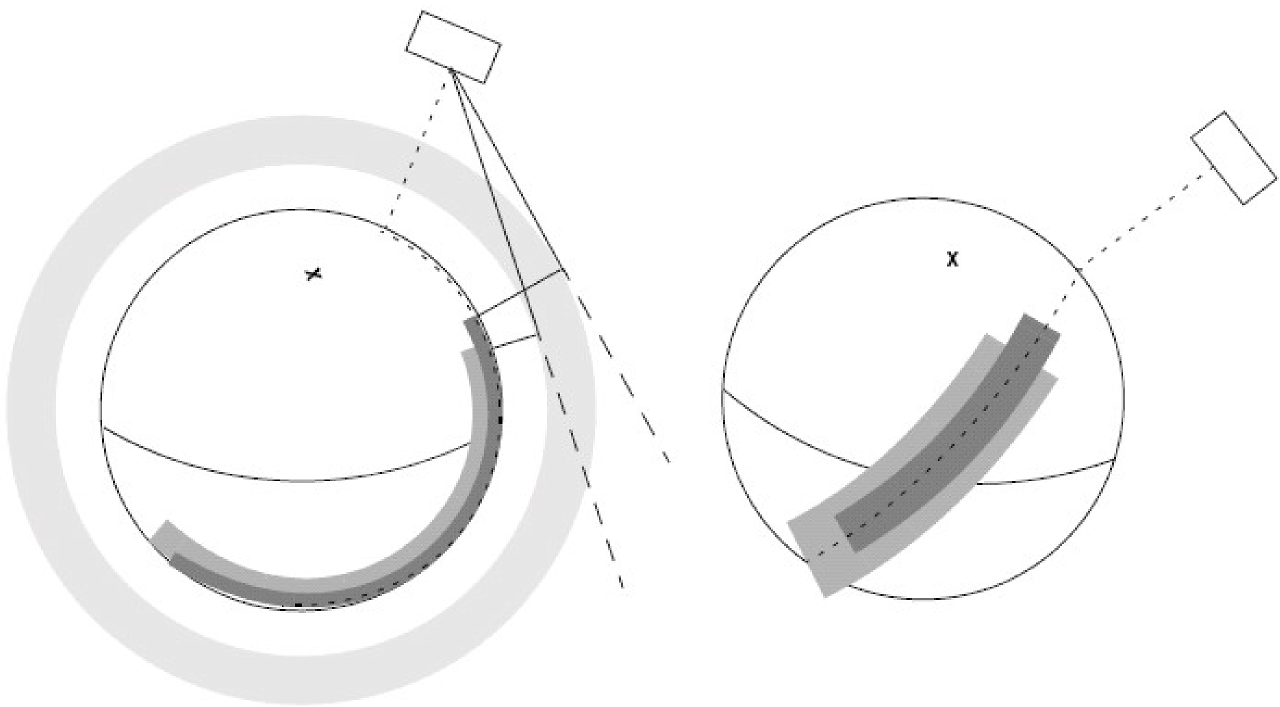

Figure 5: Limb-sounding instrument: swath definition (2)

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 3: Envisat Swaths

instrument	Mode	File Prefix = swath	Swath geometry (Table 72)	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)

7.1.3 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 4);

- 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 4: 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).

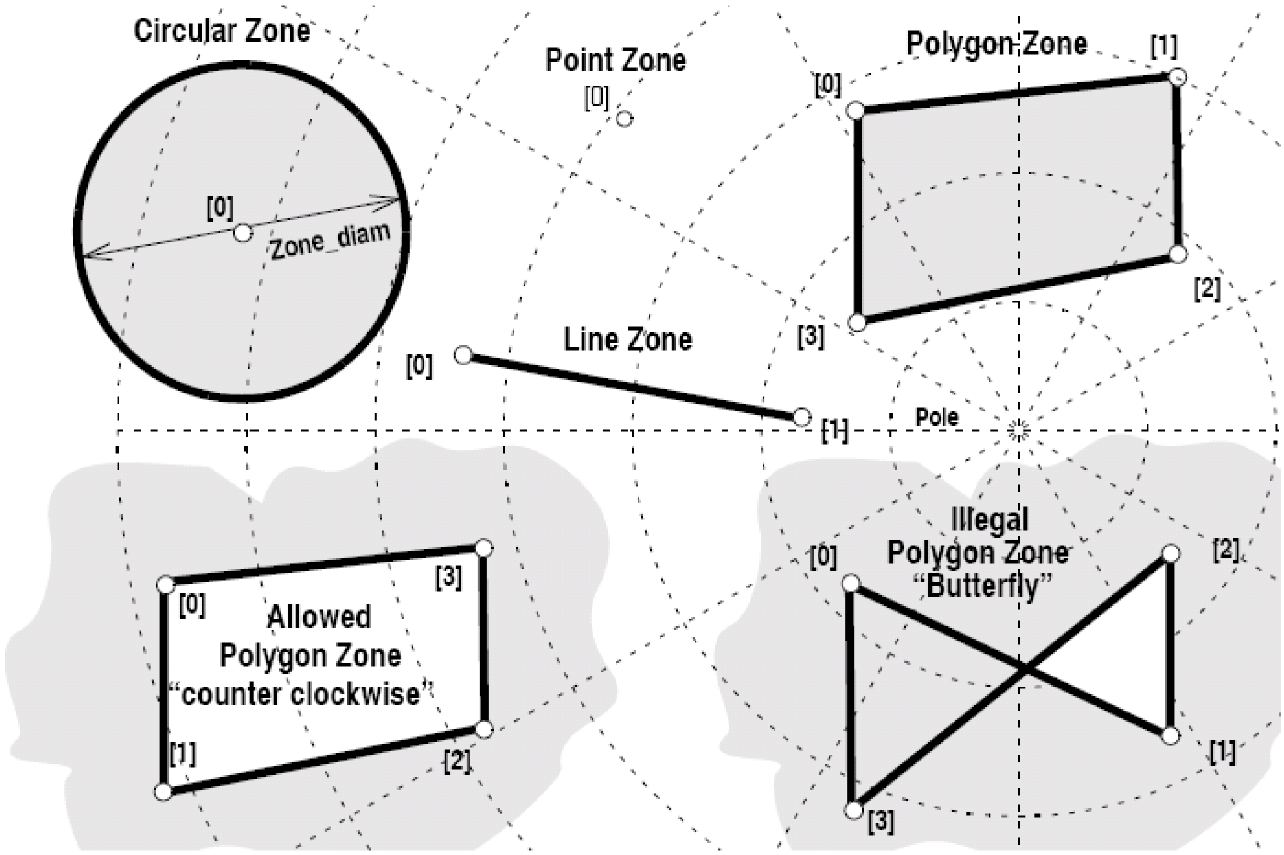

Figure 6: Zone examples

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.

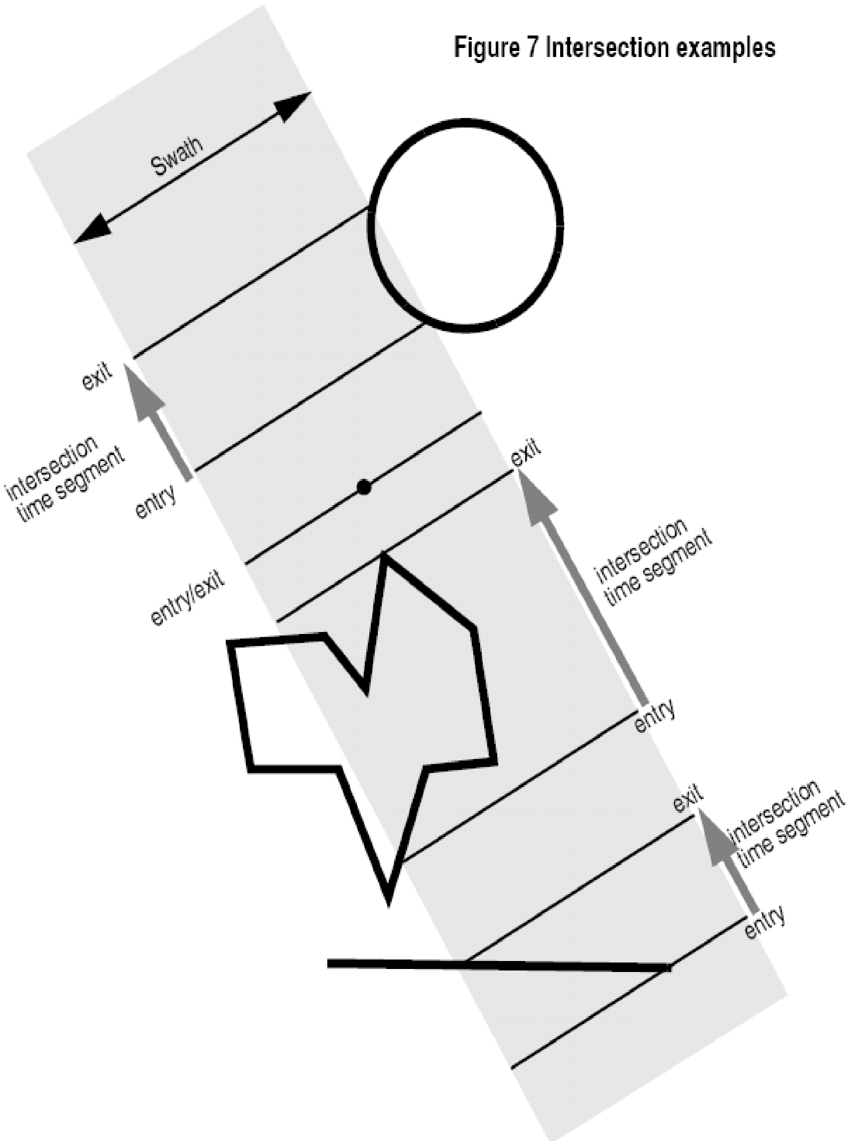

Figure 7: Intersection examples

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.

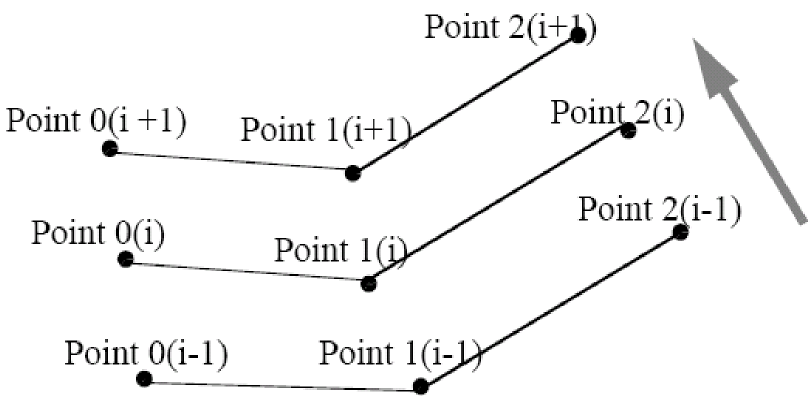

Figure 8: Swath points

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);

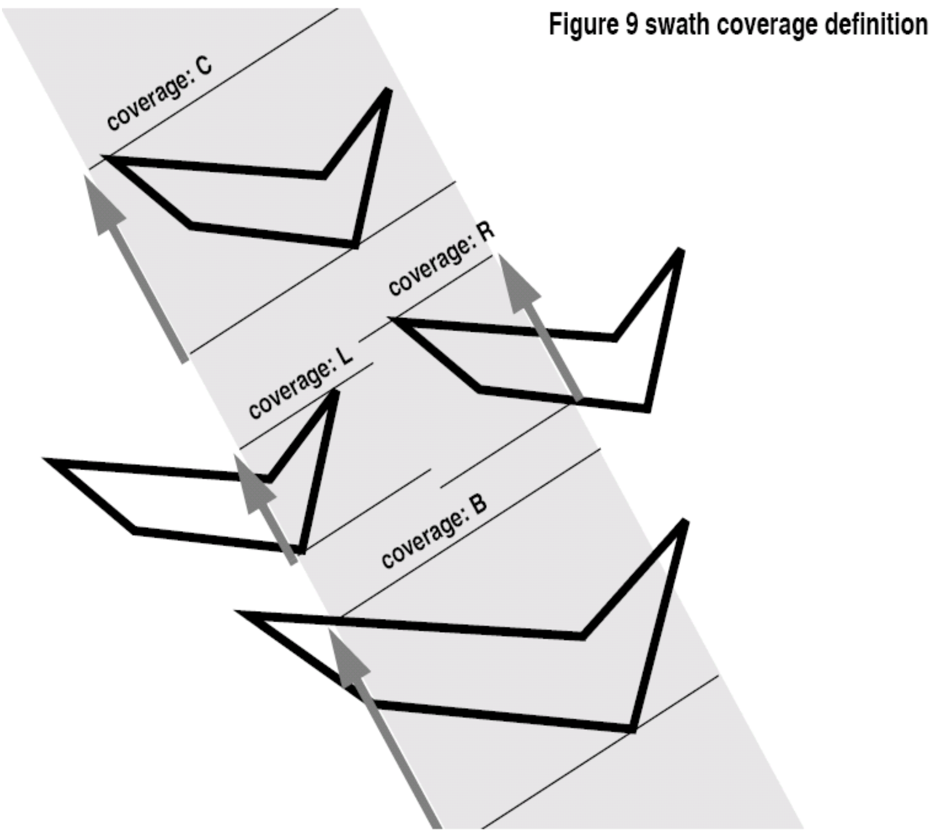

Figure 9: Swath coverage definition

- 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 5: 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 6: 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 7: 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	29
ERR	Error computing intersection of two segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SEGMENT_INTERSECT_COMP_ERR	30
ERR	Error computing gnomonic coordinates.	Computation not performed	XV_CFI_ZONE_VIS_TIME_GNOMONIC_COORD_ERR	31
ERR	Two ZONE segments intersect.	Computation not performed	XV_CFI_ZONE_VIS_TIME_TWO_SEGMENTS_INTERSECT_ERR	32
ERR	Two consecutive ZONE segments are aligned in the same direction.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ALLIGNED_SEGMENTS_ERR	33
ERR	Input parameter "ZONE diameter" cannot be negative (POINT or CIRCLE zone).	Computation not performed	XV_CFI_ZONE_VIS_TIME_ZONE_DIAM_NEGATIVE_ERR	34
ERR	SWATH contains the POLE (RECTANGULAR projection).	Computation not performed	XV_CFI_ZONE_VIS_TIME_POLE_IN_SWATH_ERR	35
ERR	Not convex SWATH quadrilateral for the specified latitude range.	Computation not performed	XV_CFI_ZONE_VIS_TIME_CUADRILATERAL_NOT_CONVEX_ERR	36
ERR	Error checking if a point is inside a quadrilateral.	Computation not performed	XV_CFI_ZONE_VIS_TIME_POINT_IN_CUADRILATERAL_ERR	37
ERR	Error sorting intersections.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SORT_INTERSECTIONS_ERR	38

ERR	Cannot (re)allocate memory for the segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SEGMENTS_MEMORY_ERR	39
ERR	Too many time segments (more than MAX_ORBITS).	Computation not performed	XV_CFI_ZONE_VIS_TIME_MAX_ORBITS_ER R	40
ERR	Cannot allocate memory for the coverage.	Computation not performed	XV_CFI_ZONE_VIS_TIME_COVERAGE_MEMORY_ERR	41
WAR N	Warning checking the visibility segments.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_CHECK_SEGMENTS_WARN	42
ERR	Error checking the visibility segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_CHECK_SEGMENTS_ERR	43
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	44
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_MODEL_E RR	45
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	46
ERR	Error computing the ANX longitude	Computation not performed	XV_CFI_ZONE_VIS_TIME_COMPUTE_ANX_ ERR	47
ERR	Error calling "orbit info"	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_INFO_ERR	48
ERR	Error computing Multi- Point swath visibilities	Computation not performed	XV_CFI_ZONE_VIS_TIME_MULTI_POINT_SWATH_INTERS_ERR	49
ERR	Error computing Point swath visibilities	Computation not performed	XV_CFI_ZONE_VIS_TIME_POINT_SWATH_INTERS_ERR	50
ERR	Error checking visibility segments	Computation not performed	XV_CFI_ZONE_VIS_TIME_ON_OFF_CHECKING_ERR	51
ERR	Error merging visibility segments	Computation not performed	XV_CFI_ZONE_VIS_TIME_MERGE_SWATH_SEGMENTS_VISIBILITIES_ERR	52
ERR	Error trying to allocate memory	Computation not performed	XV_CFI_ZONE_VIS_TIME_MEMORY_ALLOCATION_ERR	53
ERR	Error calling "swath_pos"	Computation not performed	XV_CFI_ZONE_VIS_TI	54

			ME_SWATH_POS_ER R	
ERR	Error calling "Polygon_inner_point"	Computation not performed	XV_CFI_ZONE_VIS_TIME_POLYGON_INNER_POINT_ERR	55
ERR	Error comparing orbits orbital changes	Computation not performed	XV_CFI_ZONE_VIS_TIME_CHECK_ORBITAL_CHANGE_ERR	56
ERR	Error converting zone point arrays to zone record	Computation not performed	XV_CFI_ZONE_VIS_TIME_CONVERT_ZONE_ERR	57
ERR	No suitable zone found for orbit interval	Computation not performed	XV_CFI_ZONE_VIS_TIME_ZONE_ORBIT_ERR	58
ERR	Error cloning zone	Computation not performed	XV_CFI_ZONE_VIS_TIME_CLONE_ZONE_ERR	59
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_INTERVAL_STF_ERR	60
WARN	Input orbit interval is partially outside STF validity interval	Computation performed	XV_CFI_ZONE_VIS_TIME_ORBIT_INTERVAL_STF_WARN	61
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	62
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation performed	XV_CFI_ZONE_VIS_TIME_SWATH_FLAG_LARGER_THAN_LIN_APPROX_VAL_WARN	63

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

7.1.12 Runtime performances

The following runtime performance has been measured over an interval of 50 orbits.

Table 8: Runtime performances of xv_zone_vis_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
581	253	253	39

7.2 xv_zone_vis_time_no_file

7.2.1 Overview

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 9: 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 10: 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

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_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 7) .

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 1).

7.2.6 Runtime performances

The following runtime performance has been measured over an interval of 50 orbits.

Table 11: Runtime performances of xv_zone_vis_time_no_file function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
79.6	28.6	27.6	5.2

7.3 xv_station_vis_time

7.3.1 Overview

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
- 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 69). 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.

7.3.2 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);
}
```

7.3.3 Input parameters

Table 12: 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

7.3.4 Output parameters

Table 13: 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.

7.3.5 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 14: 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_DOPPLER_ERR	
WARN	Segment longer than half nodal period deleted.	Computation performed. Message to inform the user.	XV_CFI_STATION_VIS_TIME_LONG_SEGM_SKIPPED_WARN	26
ERR	Error transforming from absolute to relative.	Computation not performed	XV_CFI_STATION_VIS_TIME_ABS_TO_REL_ERR	27
ERR	Error finding the spacecraft for the station when mask data given from file	Computation not performed	XV_CFI_STATION_VIS_TIME_MASK_FROM_FILE_NO_SC_ERR	28
ERR	Error in the mask type read from the mask data given in the file	Computation not perfomed	XV_CFI_STATION_VIS_TIME_MASK_FROM_FILE_MASK_TYPE_ERR	29
ERR	Error converting zone point array to zone record	Computation not perfomed	XV_CFI_STATION_VIS_TIME_CONVERT_ZONE_ERR	30
ERR	Input orbit interval is completely outside STF validity interval	Computation not perfomed	XV_CFI_STATION_VIS_TIME_ORBIT_INTERVAL_STF_ERR	31
WARN	Input orbit interval is partially outside STF validity interval	Computation perfomed	XV_CFI_STATION_VIS_TIME_ORBIT_INTERVAL_STF_WARN	32
ERR	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation not perfomed	XV_CFI_STATION_VIS_TIME_OSF_NON_LIN_STF_OLD_WARN	33
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation perfomed	XV_CFI_STATION_VIS_TIME_SWATH_FLAG_LARGER_THAN_LIN_APPROX_VAL_WARN	34

7.3.6 Runtime performances

The following runtime performance has been measured over an interval of 10 orbits.

Table 15: Runtime performances of xv_station_vis_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
830	352	358	58

7.4 xv_station_vis_time_no_file

7.4.1 Overview

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).

7.4.2 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);
    }
```

7.4.3 Input parameters

Table 16: 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

7.4.4 Output parameters

Table 17: 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		

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_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.

7.4.5 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 14) .

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 1).

7.4.6 Runtime performances

The following runtime performance has been measured over an interval of 10 orbits.

Table 18: Runtime performances of xv_station_vis_time_no_file function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
158.2	52.6	54.6	12.4

7.5 xv_drs_vis_time

7.5.1 Overview

The **xv_drs_vis_time** function computes all the orbital segments for which the satellite is visible from a data relay satellite located in a geostationary orbit.

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_drs_vis_time requires access to requires access to the orbit_id (xo_orbit_id) data structure. This structure can be initialized using one of the following set of 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 time intervals used by **xv_drs_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 “Data Relay 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 EO_VISIBILITY (**xo_time_to_orbit** and **xv_orbit_to_time** functions).

It is assumed that the DRS orbit has zero inclination.

The **xv_drs_vis_time** function considers the following sources of occultation:

- Earth plus 20 km of atmosphere
  - Satellite dependant sources (Currently, only Envisat model is implemented):
    - Fixed appendages: 1 deg half cone around:
    - Service Module
    - Payload Module

- Module Interface
- ASAR antenna
- AATSR Payload
- ATSR Radiator
- Mipas Payload
- Mipas Electronics
- Sciamachy Radiators A, B and C
- UMI
- Star Trackers, enlarged to have a 16 deg halfcone to protect against radiation.
- S Band Antennas
- Rotating appendices (solar array and its structure): 1 deg half cone around solar array and supporting structure
- Azimuth Blockage (165 deg to 195 deg, MCD convenction for the azimuth and elevation angles)
- Elevation Blockage (-86 deg to -90 deg, MCD convenction for the azimuth and elevation angles)

Operations of the antenna are also limited to the values (APM definition):

- Elevation from -30.0 deg to +90.0 deg
- Azimuth from -165.0 deg to +165.0 deg

These operations limitations are imposed considering margins of 1.0 deg.

In addition to these occultation sources, the function **xv_drs_vis_time** checks that the initial movement of the antenna (start-up trajectory) does not violate any mechanical constraints in order to reach the corresponding pointing to the DRS at the beginning time of the visibility segment. Similar computations are performed to be able to stop the antenna at the end point of the visibility segment.

In case the mechanical constraints are violated for a visibility segment, it is reduced by 1 second and the condition is checked again. The process is repeated until both trajectories are within the limits. A warning message is raised if the visibility segment duration comes to be smaller than the minimum duration defined by the user (*min_duration*).

The considerations assumed in the implementation of the start-up and stop trajectories are the following:

Table 19: Assumptions for the start-up and stop trajectory computations

Concept	Start-up Trajectory	Stop Trajectory
Angular movements	Common time for azimuth and elevation movement	No common time for azimuth and elevation movement
Azimuth acceleration	AZ_{acc} = 0.015 deg/sec^2	Low Velocity: AZ_{acc} = 0.023 deg/sec^2
		High Velocity: AZ_{acc} = 0.043 deg/sec^2
Elevation acceleration	EL_{acc} = 0.004 deg/sec^2	Low Velocity: EL_{acc} = 0.02 deg/sec^2
		High Velocity: EL_{acc} = 0.02 deg/sec^2
Velocity limit	N/A	vel_{limit} = 0.11459 deg/sec

7.5.2 Calling interface

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

```
#include"explorer_visibility.h"
{
      xo_orbit_id          orbit_id = {NULL};
      xp_sat_nom_trans_id sat_nom_trans_id = {NULL};
      xp_sat_trans_id      sat_trans_id = {NULL};
      xp_instr_trans_id    instr_trans_id = {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_DRS_VIS_TIME],
                    status;
      double        min_duration, longitude;

      status = xv_drs_vis_time(
                         &orbit_id, &sat_nom_trans_id,
                         &sat_trans_id, &instr_trans_id, &orbit_type,
                         &start_orbit, &start_cycle,
                         &stop_orbit, &stop_cycle,
                         &longitude, &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_id;

      status = xv_drs_vis_time_run(
                         &run_id, &orbit_type,
                         &start_orbit, &start_cycle,
                         &stop_orbit, &stop_cycle,
```

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

7.5.3 Input parameters

Table 20: Input parameters of xv_drs_vis_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
sat_nom_trans_id	xp_sat_nom_trans_id*	-	Structure that contains the Instr. Trans.	-	-
sat_trans_id	xp_sat_trans_id*	-	Structure that contains the Instr. Trans.	-	-
instr_trans_id	xp_instr_trans_id*	-	Structure that contains the Instr. Trans.	-	-
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
longitude	double		longitude of data relay satellite		[0, 360]
min_duration	double		Minimum duration for segments. Only segments with a duration longer than min_duration will be given on output.	s	≥0.0

7.5.4 Output parameters

Table 21: Output parameters of xv_drs_vis_time function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
xv_drs_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_ERR_DRS_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_drs_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.

7.5.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xv_drs_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_drs_vis_time** CFI function by calling the function of the EO_VISIBILITY software library **xv_get_code**.

Table 22: Error messages of xv_drs_vis_time

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input orbit Id.	Computation not performed	XV_CFI_DRS_VIS_TIME_ORBIT_STATUS_ ERR	0
WAR N	Input "start_orbit" below first OSF orbit: take first OSF orbit for computations.	Computation performed Message to inform the user	XV_CFI_DRS_VIS_TIME_START_ORBIT_WARN	1
ERR	Input "start_orbit" after last OSF orbit.	Computation not performed	XV_CFI_DRS_VIS_TIME_START_ORBIT_OUT_OSF_ERR	2
WAR N	Input "stop_orbit" after last OSF orbit: take last OSF orbit for computations.	Computation performed Message to inform the user	XV_CFI_DRS_VIS_TIME_STOP_ORBIT_WA RN	3
ERR	Maximum number of iterations. Orbit no: (%ld).	Computation performed Message to inform the user	XV_CFI_DRS_VIS_TIME_MAX_NUMBER_ITER_ERR	4
ERR	Input "stop_orbit" below first OSF orbit.	Computation not performed	XV_CFI_DRS_VIS_TIME_STOP_ORBIT_OUT_OSF_ERR	5
ERR	Error allocating internal memory.	Computation performed Message to inform the user	XV_CFI_DRS_VIS_TIME_INTERNAL_MEMORY_ERR	6
ERR	Error allocating memory for the time segments.	Computation not performed	XV_CFI_DRS_VIS_TIME_SEGMENTS_MEMORY_ERR	7
ERR		Computation not performed	XV_CFI_DRS_VIS_TIME_ORBIT_TYPE_ER R	8
ERR	Error in absolute start orbit computation.	Computation not performed	XV_CFI_DRS_VIS_TIME_REL_TO_ABS_START_ERR	9

ERR	Error in absolute stop orbit computation.	Computation not performed	XV_CFI_DRS_VIS_TIME_REL_TO_ABS_STOP_ERR	10
ERR	Error performing a time transformation.	Computation not performed	XV_CFI_DRS_VIS_TIME_TIME_CHANGE_E RR	11
ERR	Error transforming from TAI to TDB time.	Computation not performed	XV_CFI_DRS_VIS_TIME_TAI_TO_TDB_ER R	12
ERR	Error in XL_Sun computation.	Computation not performed	XV_CFI_DRS_VIS_TIME_XL_SUN_ERR	13
ERR	Error in Sun direction computation. Orbit no: (%ld). [PL]	Computation not performed	XV_CFI_DRS_VIS_TIME_DIR_SUN_ERR	14
ERR	Error in XL_Pt_Dir_Range computation. Orbit no: (%ld). [XL]	Computation not performed	XV_CFI_DRS_VIS_TIME_XL_PT_DIR_RANGE_ERR	15
WAR N	Error in physical mask checking. Orbit no: (%ld).	Computation performed Message to inform the user	XV_CFI_DRS_VIS_TIME_XV_CFI_FIXED_CHECK_ERR	16
ERR	Error in Earth occultation checking. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_EARTH_CHECK_ERR	17
ERR	Error in solar panel position computation. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_ROTATING_POS_ERR	18
ERR	Error in solar panel occultation checking. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_ROTATING_SOLAR_PANEL_CHECK_ERR	19
ERR	Error in solar panel structure occultation checking. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_ROTATING_SOLAR_PANEL_STR_CHECK_ERR	20
ERR	Error in OSF reading.	Computation not performed	XV_CFI_DRS_VIS_TIME_XO_LOAD_GLOBAL_OSF_ERR	21
ERR	Error in input parameters.	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_DRSINPUTS_CHECK_ERR	22
ERR	Error in canonical position computation. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_CANON_POS_ERR	23
ERR	Error in orbit parameters computation. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TIME_XV_CFI_ORBIT_INFO_ERR	24
ERR	Error in ascending node	Computation not performed	XV_CFI_DRS_VIS_TI	25

	parameters computation. Orbit no: (%ld). [PG]		ME_XO_GENSTATE_E RR	
ERR	Error in time computations. Orbit no: (%ld).	Computation not performed	XV_DRS_VIS_TIME_X V_TIME_SEC_ERR	26
WAR N	First orbit starts with visibility.	Computation performed Message to inform the user	XV_DRS_VIS_TIME_F IRST_ORBIT_VIS_WA RN	27
ERR	Last orbit ends with visibility.	Computation not performed	XV_DRS_VIS_TIME_L AST_ORBIT_VIS_WA RN	28
ERR	Error in rectifying Earth rotation. Orbit no: (%ld). [PG]	Computation not performed	XV_CFI_DRS_VIS_TI ME_XL_EF_TO_QEF_ ERR	29
ERR	Error in coordinates transformation. Orbit no: (%ld). [PL]	Computation not performed	XV_CFI_DRS_VIS_TI ME_XL_CHANGE_CS_ ERR	30
ERR	Error in azimuth- elevation computation. Orbit no: (%ld).	Computation not performed	XV_CFI_DRS_VIS_TI ME_XV_CFI_AZIM_EL EV_ERR	31
ERR	Error transforming absolute to relative begin segments.	Computation not performed	XV_CFI_DRS_VIS_TI ME_ABS_TO_REL_BG N_ERR	32
ERR	Error transforming absolute to relative end segments.	Computation not performed	XV_CFI_DRS_VIS_TI ME_ABS_TO_REL_EN D_ERR	33
ERR	Error in antenna stop trajectory computations. Orbit no: %ld.	Computation not performed	XV_DRS_VIS_TIME_X V_CHECK_STOP_TRA JECTORY_ERR	34
WAR N	No possible stop trajectory. Orbit no: %ld.	Computation performed Message to inform the user	XV_DRS_VIS_TIME_X V_CHECK_STOP_TRA JECTORY_WARN	35
ERR	Error in antenna start-up trajectory computations. Orbit no: %ld.	Computation not performed	XV_DRS_VIS_TIME_X V_CHECK_STARTUP_ TRAJECTORY_ERR	36
WAR N	No possible start-up trajectory. Orbit no: %ld.	Computation performed Message to inform the user	XV_DRS_VIS_TIME_X V_CHECK_STARTUP_ TRAJECTORY_WARN	37
ERR	Error while computing OSV (propagation/ interpolation error)	Computation not performed	XV_CFI_DRS_VIS_TI ME_OSV_COMP_ERR	38

7.5.6 Runtime performances

The following runtime performance has been measured over an interval of 10 orbits.

Table 23: Runtime performances of xv_drs_vis_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
1469	448	578	127

7.6 xv_swath_pos

7.6.1 Overview

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)
  - 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.

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

7.6.2 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);
}
```

7.6.3 Input parameters xv_swath_pos

Table 24: 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

7.6.4 Output parameters xv_swath_pos

Table 25: 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		

7.6.5 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 26: 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_SWATH_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 reading OSF information.	Computation not performed	XV_CFI_SWATH_POS _XV_OSF_RECORDS_ READ_ERR	16
ERR	Error while computing information of the orbit.	Computation not performed	XV_CFI_SWATH_POS _XV_ORBIT_INFO_ER R	17
ERR	The swath template structure contains invalid data	Computation not performed	XV_CFI_SWATH_POS _SWATH_INIT_ERR	18
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_SWATH_POS _MEMORY_ERR	19

7.6.6 Runtime performances

The following runtime performance has been measured.

Table 27: Runtime performances of xv_swath_pos function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.98	0.25	0.31	0.11

7.7 xv_star_vis_time

7.7.1 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
- 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).

7.7.2 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).

7.7.2.1 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

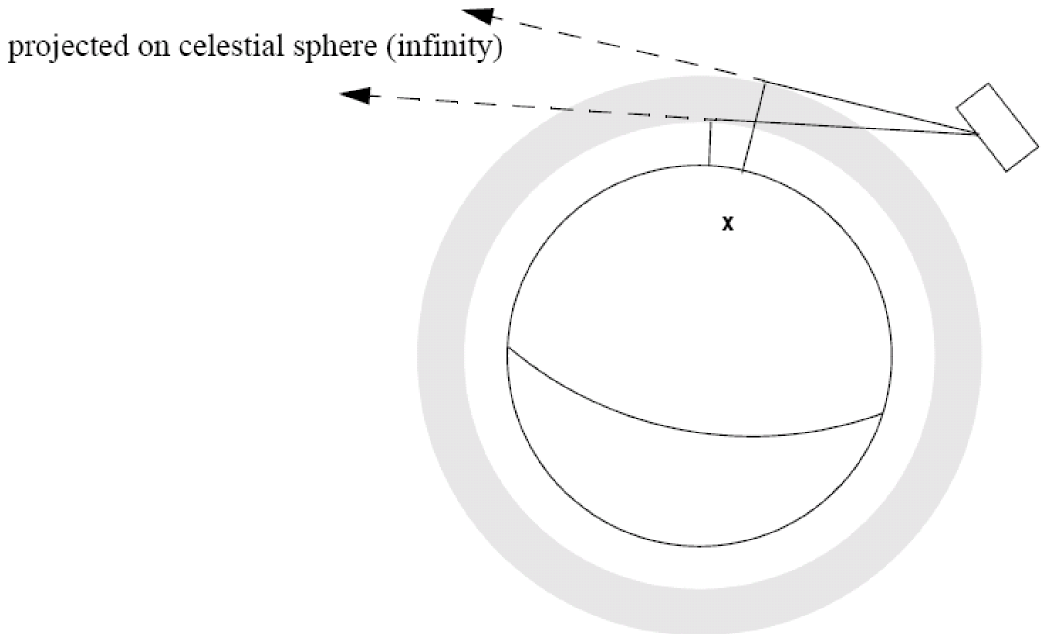

Figure 10: Two tangent altitudes over the ellipsoid

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.

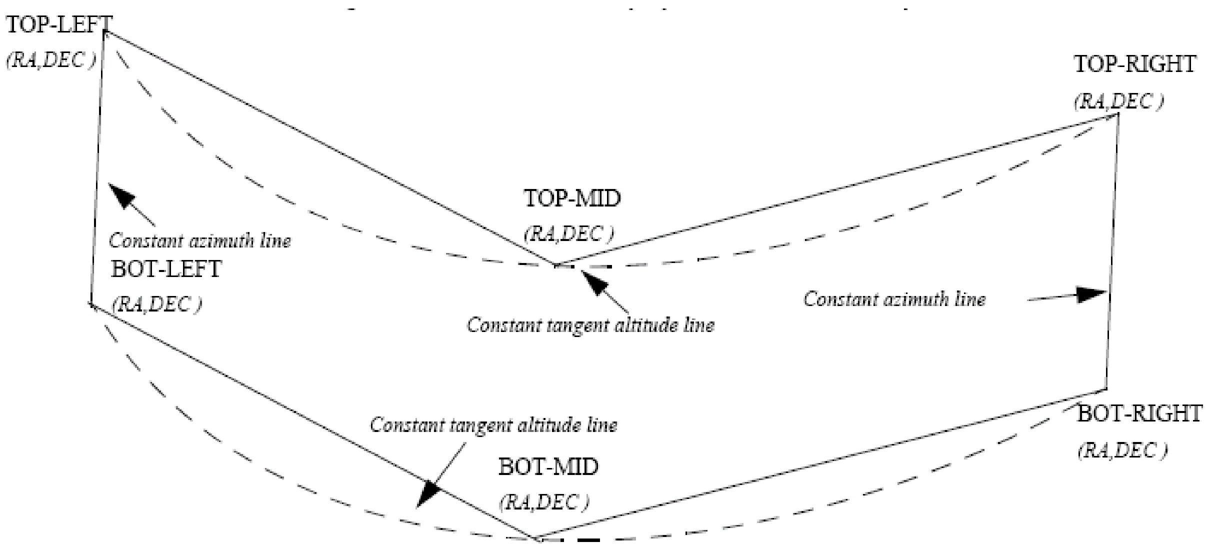

Figure 11: Instantaneous FOV projected on the celestial sphere

7.7.2.2 Splitting swaths

As it was shown in Figure 11, 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.

7.7.2.3 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.

7.7.2.4 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.

7.7.2.5 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 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.

7.7.3 Calling sequence xv_star_vis_time

For C programs, the call to **xv_star_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,
                   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);
}
```

7.7.4 Input parameters xv_star_vis_time

Table 28: 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:	-	XV_STF = 0

			• 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_SDF = 1 > 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

7.7.5 Output parameters xv_star_vis_time

Table 29: 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.

7.7.6 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 30: 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_MEMOR Y_ERR	
ERR	Error allocating memory for the coverage.	Computation not performed	XV_CFI_STAR_VIS_TIM E_COVERAGE_MEMOR Y_ERR	11
ERR	Input parameter "orbit_type" is out of range.	Computation not performed	XV_CFI_STAR_VIS_TIM E_ORBIT_TYPE_ERR	12
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_STAR_VIS_TIM E_ORBIT_MODEL_ERR	13
ERR	Error in input parameter to starvistime.	Computation not performed	XV_CFI_STAR_VIS_TIM E_INPUTS_CHECK_ERR	14
ERR	Error while transforming into absolute orbit the start_orbit.	Computation not performed	XV_CFI_STAR_VIS_TIM E_REL_TO_ABS_START _ERR	15
ERR	Error while transforming into absolute orbit the stop_orbit.	Computation not performed	XV_CFI_STAR_VIS_TIM E_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_TIM E_STAR_RADEC_ERR	17
ERR	Error obtaining orbital information.	Computation not performed	XV_CFI_STAR_VIS_TIM E_ORBIT_INFO_ERR	18
ERR	Error reading the upper swath template file.	Computation not performed	XV_CFI_STAR_VIS_TIM E_SWATH_UPPER_REA D_ERR	19
ERR	Error reading the lower swath template file.	Computation not performed	XV_CFI_STAR_VIS_TIM E_SWATH_LOWER_RE AD_ERR	20
ERR	Error reading the star data file.	Computation not performed	XV_CFI_STAR_VIS_TIM E_READ_STAR_ERR	21
ERR	Error determining transitions.	Computation not performed	XV_CFI_STAR_VIS_TIM E_STAR_MAIN_ERR	22
ERR	Error while transforming into relative orbits the output segments.	Computation not performed	XV_CFI_STAR_VIS_TIM E_ABS_TO_REL_ERR	23
ERR	Error transforming orbit to time.	Computation not performed	XV_CFI_STAR_VIS_TIM E_ORBIT_TO_TIME_ER R	24
ERR	Error reading the swath definition file: %s	Computation not performed	XV_CFI_STAR_VIS_TIM E_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	28
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	29
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	30

7.7.7 Runtime performances

The following runtime performance has been measured over an interval of 100 orbits.

Table 31: Runtime performances of xv_star_vis_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
3166	1140	1158	220

7.8 xv_multizones_vis_time

7.8.1 Overview

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 12).

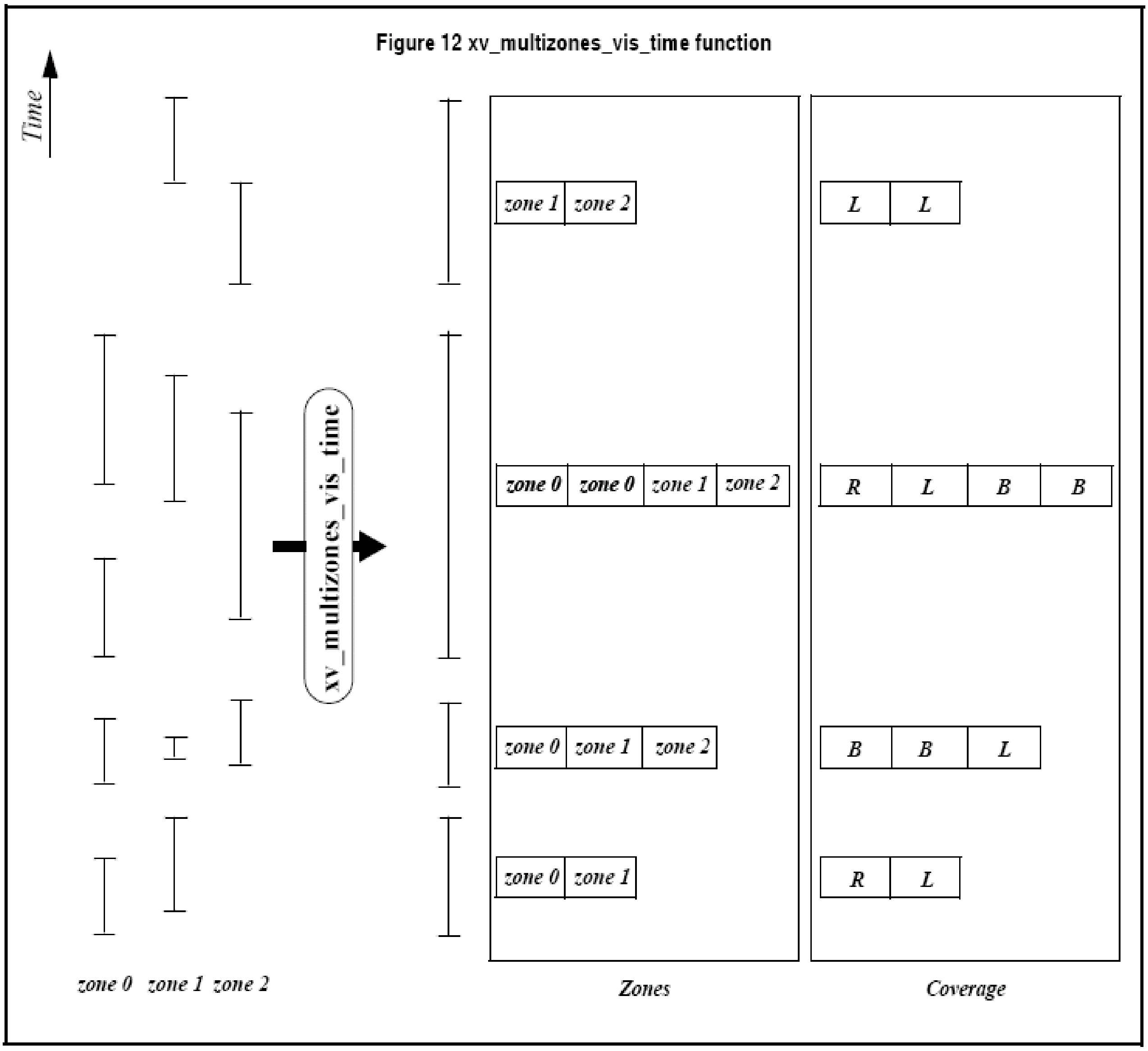

Figure 12: xv_multizones_vis_time function

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
- 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.

7.8.2 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);
}
```

7.8.3 Input parameters xv_multizones_vis_time

Table 32: 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 2)
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 2 (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[2].	deg	
zone_lat	double*	all	Geodetic latitude of - circle centre, for circ. zone. - point, for point zone.	deg	

2 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]
- ...

			- 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[3]. 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)

3 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.

7.8.4 Output parameters xv_multizones_vis_time

Table 33: 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 2
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.

7.8.5 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 34: 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

7.8.6 Runtime performances

The following runtime performance has been measured over an interval of 10 orbits.

Table 35: Runtime performances of xv_multizones_vis_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
2596	1180	1224	405

7.9 xv_multistations_vis_time

7.9.1 Overview

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
- 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.

7.9.2 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);
    }
```

7.9.3 Input parameters xv_multistations_vis_time

Table 36: 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 2)
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)

7.9.4 Output parameters xv_multistations_vis_time

Table 37: 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.

7.9.5 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 38: 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

7.9.6 Runtime performances

The following runtime performance has been measured over an interval of 10 orbits.

Table 39: Runtime performances of xv_multistations_vis_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
5721	2439	2449	405

7.10 xv_orbit_extra

7.10.1 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).

7.10.2 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);
}
```

7.10.3 Input parameters xv_orbit_extra

Table 40: 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

7.10.4 Output parameters xv_orbit_extra

Table 41: 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		

Note (Memory Management): 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.

7.10.5 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 42: 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

7.10.6 Runtime performances

The following runtime performance has been measured.

Table 43: Runtime performances of xv_orbit_extra function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
324	109	166	30

7.11 xv_gps_vis_time

TBW

7.12 xv_time_segments_not

7.12.1 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_time_segments_not** function computes the compliment of a list of orbital segments (see Figure 13)

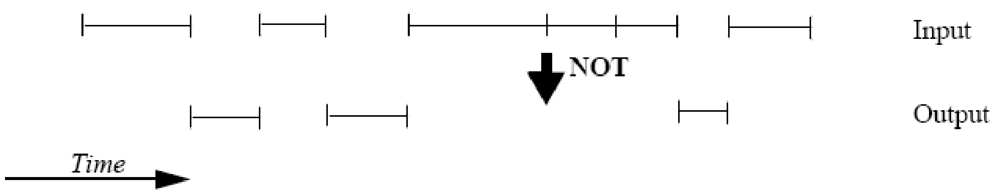

Figure 13: xv_time_segment_not_function

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.

7.12.2 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);
  }
```

7.12.3 Input parameters xv_time_segments_not

Table 44: 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 2)
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 2)
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

7.12.4 Output parameters xv_time_segments_not

Table 45: 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.

7.12.5 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 46: 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

7.12.6 Runtime performances

The following runtime performance has been measured over 34 time segments.

Table 47: Runtime performances of xv_time_segments_not function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.03	0.01	0.01	0.002

7.13 xv_time_segments_or

7.13.1 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_time_segments_or** function computes the union of a list of orbital segments (see Figure 14)

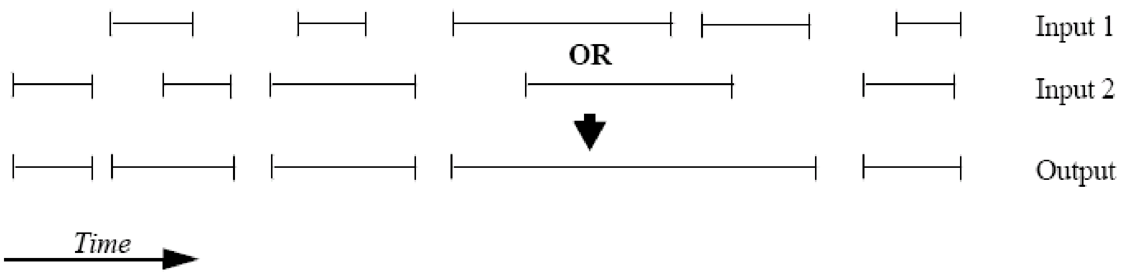

Figure 14: xv_time_segments_or_function

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.

7.13.2 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);

    }
```

7.13.3 Input parameters xv_time_segments_or

Table 48: 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 2)
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 2)
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

7.13.4 Output parameters xv_time_segments_or

Table 49: 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.

7.13.5 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 50: 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

7.13.6 Runtime performances

The following runtime performance has been measured over 34 time segments.

Table 51: Runtime performances of xv_time_segments_or function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.067	0.024	0.027	0.0045

7.14 xv_time_segments_and

7.14.1 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_time_segments_and** function computes the intersection of a list of orbital segments (see Figure 15)

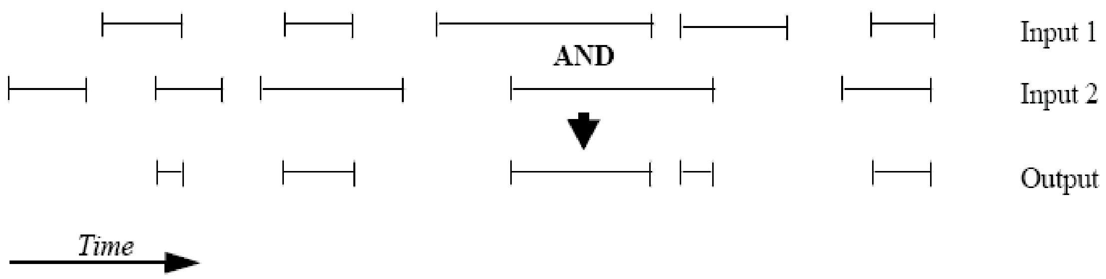

Figure 15: xv_time_segments_and_function

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.

7.14.2 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);

    }
```

7.14.3 Input parameters xv_time_segments_and

Table 52: 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 2)
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 2)
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

7.14.4 Output parameters xv_time_segments_and

Table 53: 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.

7.14.5 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 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_ AND_MEMORY_ERR	
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_REL_TO_ABS_ORBIT _ERR	
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_AND_ABS_TO_REL_ORBIT _ERR	
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_ AND_SORTING_ERR	

7.14.6 Runtime performances

The following runtime performance has been measured over 34 time segments.

Table 55: Runtime performances of xv_time_segments_and function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.11	0.041	0.043	0.0068

7.15 xv_time_segments_sort

7.15.1 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_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 56) depending on the orbit representation and the sort criteria chosen:

Table 56: 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.

7.15.2 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);
}
```

7.15.3 Input parameters xv_time_segments_sort

Table 57: 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 2)
sort_criteria	long	-	sorting criteria to be used: absolute or relative orbits	-	Complete (see Table 2)
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

7.15.4 Output parameters xv_time_segments_sort

Table 58: 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		

7.15.5 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 59: 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

7.15.6 Runtime performances

Runtime is smaller than CPU clock and it is not possible to perform loops for measuring it.

7.16 xv_time_segments_merge

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_time_segments_merge** function merges all the overlapped segments within a list (see Figure 16)

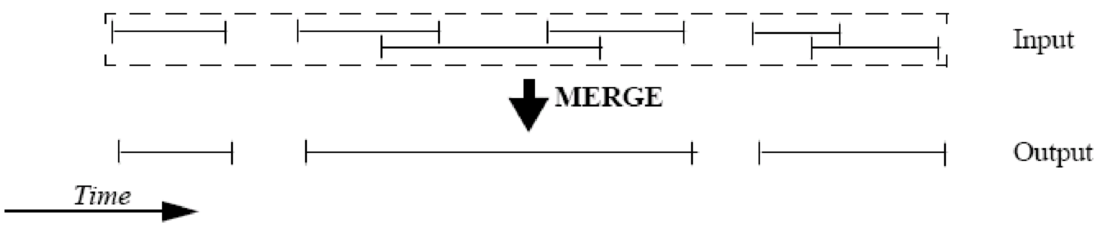

Figure 16: xv_time_segments_merge function

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.

7.16.2 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);
}
```

7.16.3 Input parameters xv_time_segments_merge

Table 60: 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 2)
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 2)
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

7.16.4 Output parameters xv_time_segments_merge

Table 61: 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.

7.16.5 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 62: 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

7.16.6 Runtime performances

The following runtime performance has been measured over 34 time segments.

Table 63: Runtime performances of xv_time_segments_merge function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.054	0.026	0.017	0.006

7.17 xv_time_segments_delta

7.17.1 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_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.

7.17.2 Calling sequence xv_time_segments_delta

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

```
#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);

}
```

7.17.3 Input parameters xv_time_segments_delta

Table 64: 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 2)
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

7.17.4 Output parameters xv_time_segments_delta

Table 65: 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.

7.17.5 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 66: 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

7.17.6 Runtime performances

The following runtime performance has been measured over 34 time segments.

Table 67: Runtime performances of xv_time_segments_delta function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
78.1	37.9	64.4	13.3

7.18 xv_time_segments_mapping

7.18.1 Overview

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 17 there are two possible different mappings:

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

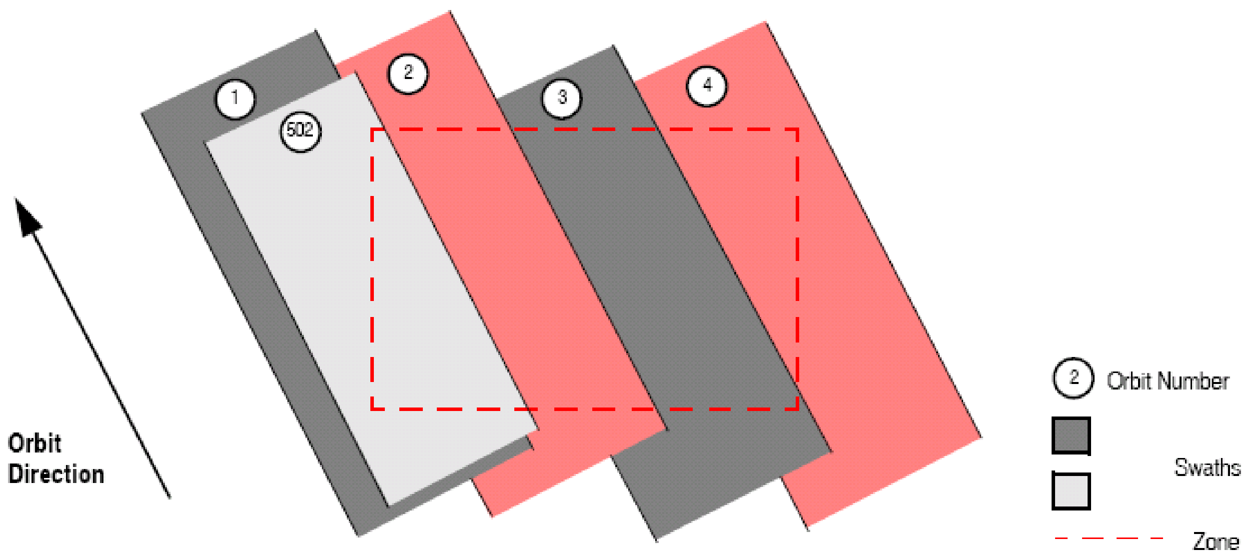

Figure 17: Different mappings with common segments

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.

7.18.2 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);
}
```

7.18.3 Input parameters xv_time_segments_mapping

Table 68: 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 2)
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		

7.18.4 Output parameters xv_time_segments_mapping

Table 69: 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 2: 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 2
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.

7.18.5 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 70: 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
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	11
ERR	Error checking extremes of the orbit range.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_CHECK_EXTREMES_ERR	12

7.18.6 Runtime performances

The following runtime performance has been measured over an interval of 50 orbits.

Table 71: Runtime performances of xv_time_segments_mapping function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
2289	1005	992	163

7.19 xv_gen_swath

7.19.1 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 72). 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 72: 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

7.19.2 Calling interface

The calling interface of the **xv_gen_swath** 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,
           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);
}
```

7.19.3 Input parameters

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

Table 73: 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 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
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	-	-

7.19.4 Output parameters

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

Table 74: 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 73)	-	-
ierr[XV_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.19.5 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 75: 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

7.19.6 Runtime performances

The following runtime performance has been measured.

Table 76: Runtime performances of xv_gen_swath function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
6517	1850	4797	590

7.19.7 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, SEOSAT, GENERIC.
- Precise propagation is used if precfile is provided.
- **Important:** gen_swath does not allow user defined atmosphere models, so the refraction model in the input SDF must be NO_REF or STD_REF.

Example:

```
gen_swath -sat ENVISAT -orbit 2000 -osf ACCEPTANCE_OSF.N1
          -sdf SDF_MERIS.1200pts.N1 -xv_v
          -dir ./gen_swath
```

7.20 xv_gen_swath_no_file

7.20.1 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.

7.20.2 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);
}
```

7.20.3 Input parameters

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

Table 77: 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	-	-

7.20.4 Output parameters

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

Table 78: 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	-	-

7.20.5 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 75) .

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 1).

7.20.6 Runtime performances

The following runtime performance has been measured.

Table 79: Runtime performances of xv_gen_swath_no_file function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
2330	1203	4080	360

7.21 xv_gen_scf

7.21.1 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).

7.21.2 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);
    }
```

7.21.3 Input parameters

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

Table 80: 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_appear	-	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	-	-

7.21.4 Output parameters

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

Table 81: 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 80)	-	-
ierr[XV_NUM_ERR_GEN_SCF]	long	all	Status vector	-	-

7.21.5 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 82: 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

7.21.6 Runtime performances

The following runtime performance has been measured for the generation of a SCF with 27 visibility segments.

Table 83: Runtime performances of xv_gen_scf function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
64	36	12	8

8 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 84: Known problems

CFI library	Problem	Work around solution
~~Fortran~~	~~No fortran version of the library exits~~	-
xv_gps_vis_time	Functions not available yet	-