

Earth Observation  
Mission CFI Software  
  
EO\_VISIBILITY  
SOFTWARE USER MANUAL

Code: EO-MA-DMS-GS-0006  
Issue: 4.5  
Date: 01/03/13

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

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

© DEIMOS Space S.L.U

All Rights Reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of DEIMOS Space S.L.U or ESA.

## DOCUMENT INFORMATION

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

External Distribution		
Name	Organisation	Copies

Electronic handling	
Word Processor:	LibreOffice 3.6
Archive Code:	P/SUM/DMS/01/026-029
Electronic file name:	eo-ma-dms-gs-006-21

## DOCUMENT STATUS LOG

Issue	Change Description	Date	Approval
1.0	Unreleased	19/06/02	
2.0	Complete document	29/11/02	
2.1	Maintenance release with the following main changes: <ul style="list-style-type: none"> <li>• xv_multizones_vis_time added.</li> <li>• xv_multistation_vis_time added.</li> <li>• xv_time_segment_mapping added.</li> <li>• xv_orbit_extra added.</li> </ul>	13/05/03	
2.2	Maintenance release	30/09/03	
2.2.2	Small interface change 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: <ul style="list-style-type: none"> <li>• Use of Predicted Orbit/Orbit event files.</li> <li>• Use of Swath Definition files.</li> </ul>	13/10/04	
3.2	Maintenance release	15/11/04	
3.3	New features: <ul style="list-style-type: none"> <li>• Use of Predicted Orbit/Orbit event files for all visibility functions.</li> <li>• Use of Swath Definition files for all visibility functions.</li> <li>• ENVISAT ASCII files are no longer supported</li> </ul>	11/07/05	
3.4	<ul style="list-style-type: none"> <li>• Maintenance release.</li> <li>• gen_swath executable moved to this library.</li> <li>• Changes in the xv_swath_pos interface</li> </ul>	18/11/05	
3.5	Maintenance release.	26/05/06	
3.6	Maintenance release.	24/11/06	

3.7	<ul style="list-style-type: none"> <li>• Maintenance release.</li> <li>• New features:           <ul style="list-style-type: none"> <li>- xv_gen_scf</li> <li>- expcfi_check_libs</li> <li>- xv_zone_vis_time_no_file</li> <li>- xv_station_vis_time_no_file</li> <li>- xv_gen_swath_no_file</li> <li>- library version for Mac OS X on Intel (32 and 64 bits)</li> </ul> </li> </ul>	13/07/07	
3.7.2	<ul style="list-style-type: none"> <li>• Maintenance release.</li> <li>• New features:           <ul style="list-style-type: none"> <li>- Curved and closed swaths for xv_zone_vis_time</li> </ul> </li> </ul>	31/07/08	
4.0	Maintenance release.	19/01/09	
4.1	<ul style="list-style-type: none"> <li>• Maintenance release</li> <li>• New features:           <ul style="list-style-type: none"> <li>- AOS/LOS mask mode from Ground Station DB 1.4 file</li> <li>- Visibility across orbital changes</li> </ul> </li> </ul>	07/05/10	
4.2	<ul style="list-style-type: none"> <li>• Maintenance release</li> <li>• New features:           <ul style="list-style-type: none"> <li>- Support of visibility functions with TLE and precise propagator</li> </ul> </li> </ul>		
4.3	<ul style="list-style-type: none"> <li>• Maintenance release</li> </ul>		
4.4	<ul style="list-style-type: none"> <li>• Maintenance release</li> <li>• New features:           <ul style="list-style-type: none"> <li>- Function xv_sc_vis_time</li> </ul> </li> </ul>	05/07/12	
4.5	<ul style="list-style-type: none"> <li>• Maintenance release</li> </ul>	01/03/13	

## TABLE OF CONTENTS

DOCUMENT INFORMATION.....	2
DOCUMENT STATUS LOG.....	3
TABLE OF CONTENTS.....	5
LIST OF TABLES.....	11
LIST OF FIGURES.....	13
1 SCOPE.....	14
2 ACRONYMS, NOMENCLATURE AND TERMINOLOGY.....	15
2.1 Acronyms.....	15
2.2 Nomenclature.....	15
2.3 Note on Terminology.....	16
3 APPLICABLE AND REFERENCE DOCUMENTS.....	17
3.1 Applicable Documents.....	17
3.2 Reference Documents.....	17
4 INTRODUCTION.....	18
4.1 Functions Overview.....	18
4.2 Calling Sequence.....	20
5 LIBRARY INSTALLATION.....	21
6 LIBRARY USAGE.....	22
6.1 Usage hints.....	24
6.2 General enumerations.....	24
6.3 Data Structures.....	25

---

<b>7 CFI FUNCTIONS DESCRIPTION.....</b>	<b>27</b>
<b>7.1 xv_zone_vis_time.....</b>	<b>28</b>
7.1.1 Overview.....	28
7.1.2 Swath Definition.....	30
7.1.2.1 Earth-observing Instruments Swath Definition.....	30
7.1.2.2 Limb-sounding Instruments Swath Definition.....	31
7.1.2.3 Limb-sounding Instruments Inertial Swath Definition.....	33
7.1.2.4 Swath Definition for Envisat.....	33
7.1.3 Zone Borders and Projection.....	35
7.1.4 Zone Definition.....	35
7.1.5 Intersection Definition.....	37
7.1.6 Intersection Algorithm.....	38
7.1.6.1 Intersection with a point swath.....	38
7.1.6.2 Intersection with a segment swath.....	38
7.1.6.3 Intersection with a multi-segment swath.....	39
7.1.7 Usage Hints.....	40
7.1.7.1 Limb-sounding Instruments Intersection.....	40
7.1.7.2 Zone Coverage.....	40
7.1.7.3 Combined use of xv_swath_pos and the coverage flag .....	41
7.1.8 Calling sequence.....	42
7.1.9 Input parameters.....	44
7.1.10 Output parameters.....	47
7.1.11 Warnings and errors.....	49
<b>7.2 xv_zone_vis_time_no_file.....</b>	<b>54</b>
7.2.1 Overview.....	54
7.2.2 Calling sequence.....	54
7.2.3 Input parameters.....	56
7.2.4 Output parameters.....	59
7.2.5 Warnings and errors.....	61
<b>7.3 xv_station_vis_time.....</b>	<b>62</b>
7.3.1 Overview.....	62
7.3.2 Calling interface.....	64
7.3.3 Input parameters.....	66

---

7.3.4	Output parameters.....	69
7.3.5	Warnings and errors.....	71
<b>7.4</b>	<b>xv_station_vis_time_no_file.....</b>	<b>74</b>
7.4.1	Overview.....	74
7.4.2	Calling interface.....	74
7.4.3	Input parameters.....	76
7.4.4	Output parameters.....	79
7.4.5	Warnings and errors.....	81
<b>7.5</b>	<b>xv_sc_vis_time.....</b>	<b>82</b>
7.5.1	Overview.....	82
7.5.2	Calling interface.....	82
7.5.3	Input parameters.....	85
7.5.4	Output parameters.....	87
7.5.5	Warnings and errors.....	89
<b>7.6</b>	<b>xv_swath_pos.....</b>	<b>91</b>
7.6.1	Overview.....	91
7.6.2	Calling sequence of xv_swath_pos.....	92
7.6.3	Input parameters xv_swath_pos.....	93
7.6.4	Output parameters xv_swath_pos.....	93
7.6.5	Warnings and errors.....	95
<b>7.7</b>	<b>xv_star_vis_time.....</b>	<b>97</b>
7.7.1	Overview.....	97
7.7.2	Swath Definition.....	98
7.7.2.1	Inertial Swaths.....	98
7.7.2.2	Splitting swaths.....	99
7.7.2.3	Orbital Changes.....	99
7.7.2.4	Format of Swath Template File.....	100
7.7.2.5	MLST non linear drift.....	100
7.7.3	Calling sequence xv_star_vis_time.....	101
7.7.4	Input parameters xv_star_vis_time.....	103
7.7.5	Output parameters xv_star_vis_time.....	105
7.7.6	Warnings and errors.....	107

---

<b>7.8 xv_multizones_vis_time.....</b>	<b>110</b>
7.8.1 Overview.....	110
7.8.2 Calling sequence xv_multizones_vis_time.....	113
7.8.3 Input parameters xv_multizones_vis_time.....	115
7.8.4 Output parameters xv_multizones_vis_time.....	118
7.8.5 Warnings and errors.....	120
<b>7.9 xv_multistations_vis_time.....</b>	<b>121</b>
7.9.1 Overview.....	121
7.9.2 Calling sequence xv_multistations_vis_time.....	123
7.9.3 Input parameters xv_multistations_vis_time.....	125
7.9.4 Output parameters xv_multistations_vis_time.....	127
7.9.5 Warnings and errors.....	129
<b>7.10 xv_orbit_extra.....</b>	<b>130</b>
7.10.1 Overview.....	130
7.10.2 Calling sequence xv_orbit_extra.....	131
7.10.3 Input parameters xv_orbit_extra.....	132
7.10.4 Output parameters xv_orbit_extra.....	133
7.10.5 Warnings and errors.....	135
<b>7.11 xv_gps_vis_time.....</b>	<b>136</b>
<b>7.12 xv_time_segments_not.....</b>	<b>137</b>
7.12.1 Overview.....	137
7.12.2 Calling sequence xv_time_segments_not.....	138
7.12.3 Input parameters xv_time_segments_not.....	140
7.12.4 Output parameters xv_time_segments_not.....	141
7.12.5 Warnings and errors.....	142
<b>7.13 xv_time_segments_or.....</b>	<b>143</b>
7.13.1 Overview.....	143
7.13.2 Calling sequence xv_time_segments_or.....	144
7.13.3 Input parameters xv_time_segments_or.....	146
7.13.4 Output parameters xv_time_segments_or.....	148
7.13.5 Warnings and errors.....	149
<b>7.14 xv_time_segments_and.....</b>	<b>150</b>



---

7.14.1 Overview.....	150
7.14.2 Calling sequence xv_time_segments_and.....	151
7.14.3 Input parameters xv_time_segments_and.....	153
7.14.4 Output parameters xv_time_segments_and.....	155
7.14.5 Warnings and errors.....	156
<b>7.15 xv_time_segments_sort.....</b>	<b>157</b>
7.15.1 Overview.....	157
7.15.2 Calling sequence xv_time_segments_sort.....	157
7.15.3 Input parameters xv_time_segments_sort.....	159
7.15.4 Output parameters xv_time_segments_sort.....	160
7.15.5 Warnings and errors.....	161
<b>7.16 xv_time_segments_merge.....</b>	<b>162</b>
7.16.1 Overview.....	162
7.16.2 Calling sequence xv_time_segments_merge.....	163
7.16.3 Input parameters xv_time_segments_merge.....	165
7.16.4 Output parameters xv_time_segments_merge.....	166
7.16.5 Warnings and errors.....	167
<b>7.17 xv_time_segments_delta.....</b>	<b>168</b>
7.17.1 Overview.....	168
7.17.2 Calling sequence xv_time_segments_delta.....	169
7.17.3 Input parameters xv_time_segments_delta.....	171
7.17.4 Output parameters xv_time_segments_delta.....	172
7.17.5 Warnings and errors.....	173
<b>7.18 xv_time_segments_mapping.....</b>	<b>174</b>
7.18.1 Overview.....	174
7.18.2 Calling sequence xv_time_segments_mapping.....	176
7.18.3 Input parameters xv_time_segments_mapping.....	178
7.18.4 Output parameters xv_time_segments_mapping.....	181
7.18.5 Warnings and errors.....	183
<b>7.19 xv_gen_swath.....</b>	<b>185</b>
7.19.1 Overview.....	185
7.19.2 Calling interface.....	186

---

7.19.3	Input parameters.....	187
7.19.4	Output parameters.....	188
7.19.5	Warnings and errors.....	189
7.19.6	Executable Program.....	190
<b>7.20</b>	<b>xv_gen_swath_no_file.....</b>	<b>192</b>
7.20.1	Overview.....	192
7.20.2	Calling interface.....	192
7.20.3	Input parameters.....	193
7.20.4	Output parameters.....	193
7.20.5	Warnings and errors.....	194
<b>7.21</b>	<b>xv_gen_scf.....</b>	<b>195</b>
7.21.1	Overview.....	195
7.21.2	Calling interface.....	195
7.21.3	Input parameters.....	196
7.21.4	Output parameters.....	197
7.21.5	Warnings and errors.....	198
<b>8</b>	<b>RUNTIME PERFORMANCES.....</b>	<b>199</b>
<b>9</b>	<b>KNOWN PROBLEMS.....</b>	<b>202</b>

## LIST OF TABLES

Table 1: CFI functions included within EO_VISIBILITY library.....	23
Table 2: Some enumerations within EO_VISIBILITY library.....	24
Table 3: EO_VISIBILITY structures.....	26
Table 4: Envisat Swaths.....	33
Table 5: Zone definition.....	35
Table 6: Input parameters of xv_zone_vis_time function.....	44
Table 7: Output parameters of xv_zone_vis_time function.....	47
Table 8: Error messages and codes for xv_zone_vis_time.....	49
Table 9: Input parameters of xv_zone_vis_time_no_file function.....	56
Table 10: Output parameters of xv_zone_vis_time_no_file function.....	59
Table 11: Input parameters of xv_station_vis_time.....	66
Table 12: Output parameters of xv_station_vis_time function.....	69
Table 13: Error messages and codes for xv_station_vis_time.....	71
Table 14: Input parameters of xv_station_vis_time_no_file.....	76
Table 15: Output parameters of xv_station_vis_time_no_file function.....	79
Table 16: Input parameters of xv_sc_vis_time.....	85
Table 17: Output parameters of xv_sc_vis_time function.....	87
Table 18: Error messages of xv_sc_vis_time.....	89
Table 19: Input parameters of xv_swath_pos.....	93
Table 20: Output parameters of xv_swath_pos.....	93
Table 21: Error messages and codes.....	95
Table 22: Input parameters of xv_star_vis_time.....	103
Table 23: Output Parameters of xv_star_vis_time.....	105
Table 24: Error messages and codes.....	107
Table 25: Input parameters of xv_multizones_vis_time.....	115
Table 26: Output parameters of xv_multizones_vis_time.....	118
Table 27: Error messages and codes.....	120
Table 28: Input parameters of xv_multistations_vis_time.....	125
Table 29: Output parameters of xv_multistations_vis_time.....	127
Table 30: Error messages and codes.....	129
Table 31: Input parameters of xv_orbit_extra.....	132
Table 32: Output parameters of xv_orbi_extra.....	133
Table 33: Error messages and codes.....	135
Table 34: Input parameters of xv_time_segments_not.....	140
Table 35: Output parameters of xv_time_segments_not.....	141
Table 36: Error messages and codes.....	142
Table 37: Input parameters of xv_time_segments_or.....	146
Table 38: Output parameters of xv_time_segments_or.....	148
Table 39: Error messages and codes.....	149
Table 40: Input parameters of xv_time_segments_and.....	153
Table 41: Output parameters of xv_time_segments_and.....	155
Table 42: Error messages and codes.....	156
Table 43: xv_time_segments_sort function.....	157
Table 44: Input parameters of xv_time_segments_sort.....	159
Table 45: Output parameters of xv_time_segments_sort.....	160
Table 46: Error messages and codes.....	161

Table 47: Input parameters of xv_time_segments_merge.....	165
Table 48: Output parameters of xv_time_segments_merge.....	166
Table 49: Error messages and codes.....	167
Table 50: Input parameters of xv_time_segments_delta.....	171
Table 51: Output parameters of xv_time_segments_delta.....	172
Table 52: Error messages and codes.....	173
Table 53: Input parameters of xv_time_segments_mapping.....	178
Table 54: Output parameters of xv_time_segments_mapping.....	181
Table 55: Error messages and codes.....	183
Table 56: Swath geometry definition (algorithm).....	185
Table 57: Input parameters of xv_gen_swath function.....	187
Table 58: Output parameters of xv_gen_swath function.....	188
Table 59: Error messages of xv_gen_swath function.....	189
Table 60: Input parameters of xv_gen_swath_no_file function.....	193
Table 61: Output parameters of xv_gen_swath_no_file function.....	193
Table 62: Input parameters of xv_gen_scf function.....	196
Table 63: Output parameters of xv_gen_scf function.....	197
Table 64: Error messages of xv_gen_scf function.....	198
Table 65: Known problems.....	202

## LIST OF FIGURES

Figure 1: EO_VISIBILITY Data Flow.....	20
Figure 2: Segment Definition xv_zone_vis_time.....	28
Figure 3: Earth-observing instrument: swath definition.....	31
Figure 4: Limb-sounding instrument: swath definition (1).....	32
Figure 5: Limb-sounding instrument: swath definition (2).....	32
Figure 6: Zone examples.....	36
Figure 7: Intersection examples.....	37
Figure 8: Swath points .....	39
Figure 9: Swath coverage definition.....	40
Figure 10: Two tangent altitudes over the ellipsoid.....	98
Figure 11: Instantaneous FOV projected on the celestial sphere.....	99
Figure 12: xv_multizones_vis_time function.....	111
Figure 13: xv_time_segment_not_function.....	137
Figure 14: xv_time_segments_or_function.....	143
Figure 15: xv_time_segments_and_function.....	150
Figure 16: xv_time_segments_merge function.....	162
Figure 17: Different mappings with common segments.....	174

## 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

<i>CFI</i>	A group of CFI functions, and related software and documentation that will be distributed by ESA to the users as an independent unit
<i>CFI function</i>	A single function within a CFI that can be called by the user
<i>Library</i>	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 4.4 05/07/12

[MSC] Earth Observation Mission CFI Software. Mission Specific Customizations Document. EO-MA-DMS-GS-0018. Issue 4.4 05/07/12.

[F\_H\_SUM] Earth Observation Mission CFI Software. EO\_FILE\_HANDLING Software User Manual. EO-MA-DMS-GS-0008. Issue 4.4 05/07/12.

[D\_H\_SUM] Earth Observation Mission CFI Software. EO\_DATA\_HANDLING Software User Manual. EO-MA-DMS-GS-0007. Issue 4.4 05/07/12.

[LIB\_SUM] Earth Observation Mission CFI Software. EO\_LIB Software User Manual. EO-MA-DMS-GS-0003. Issue 4.4 05/07/12.

[ORBIT\_SUM] Earth Observation Mission CFI Software. EO\_ORBIT Software User Manual. EO-MA-DMS-GS-0004. Issue 4.4 05/07/12.

[POINT\_SUM] Earth Observation Mission CFI Software. EO\_POINTING Software User Manual. EO-MA-DMS-GS-0005. Issue 4.4 05/07/12.

[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\_sc\_vis\_time**: computes visibility time segments for a target 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,

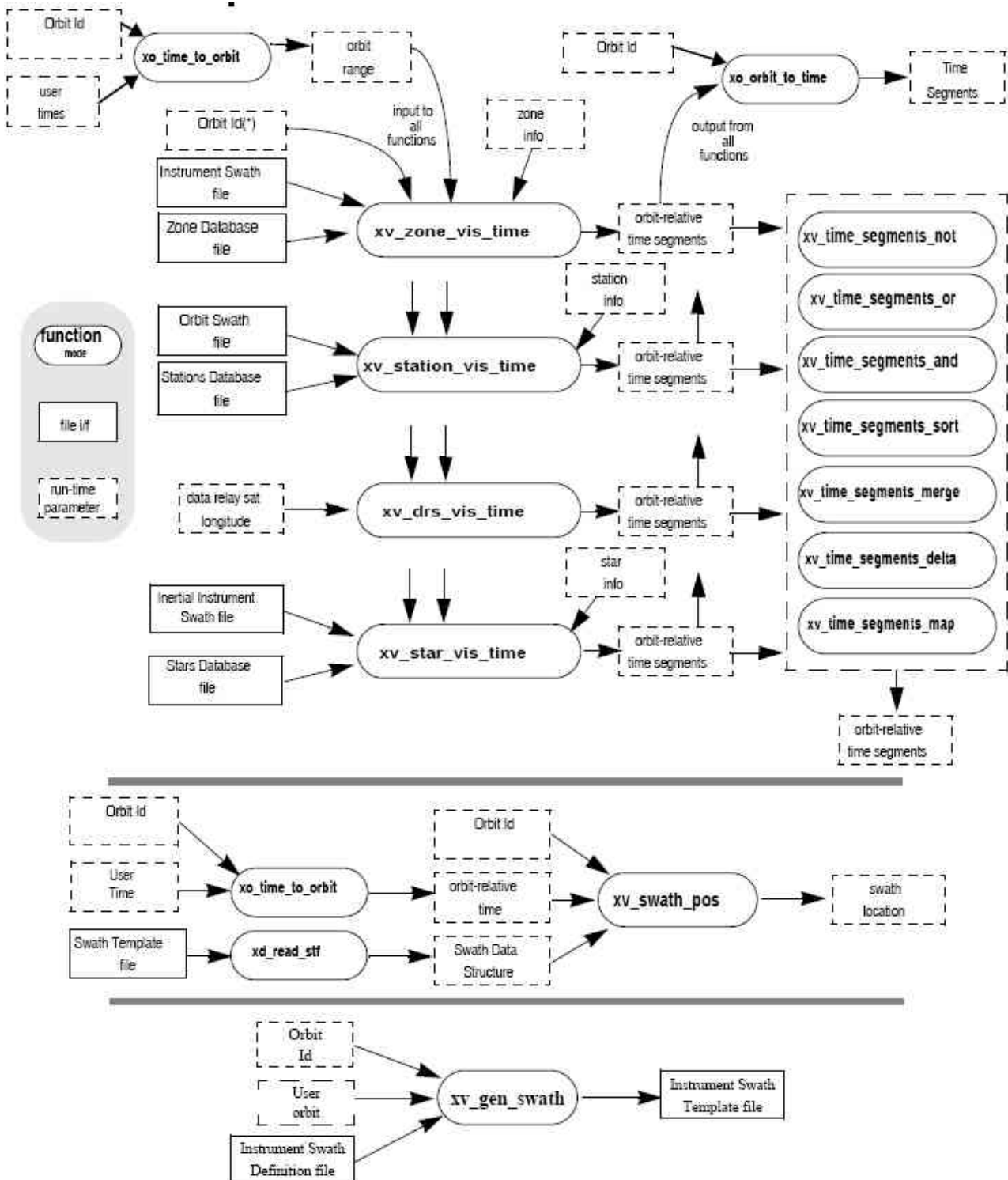


Figure 1: EO\_VISIBILITY Data Flow

\*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:

- 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)

Some functions use internally paralelization with OpenMP, so it is needed to use the proper flags in the compilation so the compiler find the needed symbols:

- WINDOWS: /openmp
- LINUX And MacOS: -fopenmp

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):

- LINUX:

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

- Windows users:

```
/openmp /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  
pthread.lib
```

- MacOS:

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

```
-lexplorer_pointing -lexplorer_orbit -lexplorer_lib
-lexplorer_data_handling -lexplorer_file_handling
-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		
<code>xv_zone_vis_time</code> <code>xv_zone_vis_time_no_file</code>	<code>XV_ZONE_VIS_TIME_ID</code>	0
<code>xv_station_vis_time</code> <code>xv_station_vis_time_no_file</code>	<code>XV_STATION_VIS_TIME_ID</code>	1
<code>xv_sc_vis_time</code>	<code>XV_SC_VIS_TIME_ID</code>	2
<code>xv_swath_pos_id</code>	<code>XV_SWATH_POS_ID</code>	3
<code>xv_star_vis_time</code>	<code>XV_STAR_VIS_TIME_ID</code>	4
<code>xv_multizones_vis_time</code>	<code>XV_MULTIZONES_VIS_TIME_ID</code>	5
<code>xv_multistations_vis_time</code>	<code>XV_MULTISTATIONS_VIS_TIME_ID</code>	6
<code>xv_time_segments_not</code>	<code>XV_TIME_SEGMENTS_NOT_ID</code>	7
<code>xv_time_segments_or</code>	<code>XV_TIME_SEGMENTS_OR_ID</code>	8
<code>xv_time_segments_and</code>	<code>XV_TIME_SEGMENTS_AND_ID</code>	9
<code>xv_time_segments_sort</code>	<code>XV_TIME_SEGMENTS_SORT_ID</code>	10
<code>xv_time_segments_merge</code>	<code>XV_TIME_SEGMENTS_MERGE_ID</code>	11
<code>xv_time_segments_delta</code>	<code>XV_TIME_SEGMENTS_DELTA_ID</code>	12
<code>xv_time_segments_mapping</code>	<code>XV_TIME_SEGMENTS_MAPPING_ID</code>	13
<code>xv_orbit_extra</code>	<code>XV_ORBIT_EXTRA_ID</code>	14
<code>xv_gen_swath</code> <code>xv_gen_swath_no_file</code>	<code>XV_GEN_SWATH_ID</code>	15
<code>xv_gen_scf</code>	<code>XV_GEN_SCF_ID</code>	16
Error Handling Functions		
<code>xv_verbose</code> <code>xv_silent</code>	not applicable	

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 coverage 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
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**.

## 6.3 Data Structures

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

**Table 3: EO\_VISIBILITY structures**

Structure name	Data		
	Variable Name	C type	Description
xv_az_el_mask	num_mask_pt	long	Number of azimuth and elevation pairs defining the antenna mask
	azimuth	double [XD_VERTICES]	Azimuth defining the antenna mask
	elevation	double [XD_VERTICES]	Elevation defining the antenna mask
xv_link_mask	incl_mask	xv_az_el_mask	List of azimuth and elevation pairs in Instrument Frame defining an inclusive zone
	excl_mask	xv_az_el_mask	List of azimuth and elevation pairs in Instrument Frame defining an exclusive zone
xv_link_data	mask_data	xv_link_mask	List of azimuth and elevation pairs in Instrument Frame
	min_tg_height	double	Minimum tangent height

## 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

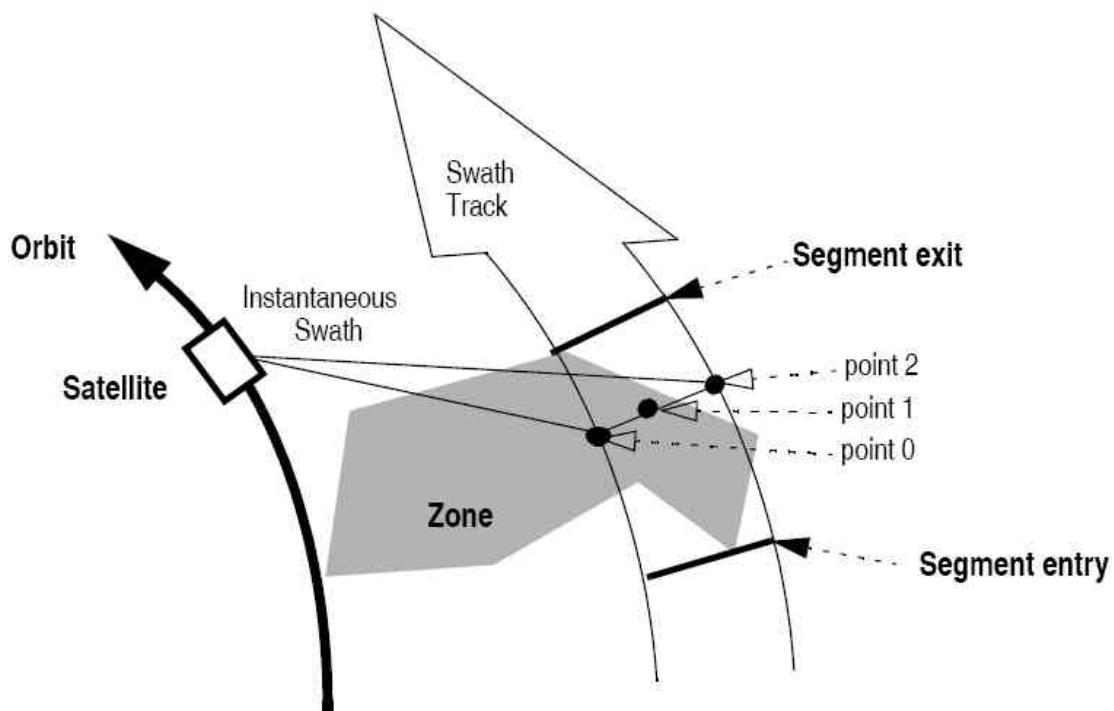


Figure 2: Segment Definition `xv_zone_vis_time`

`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\_Vailability\_Range* and *Stop\_Vailability\_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 44). 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:

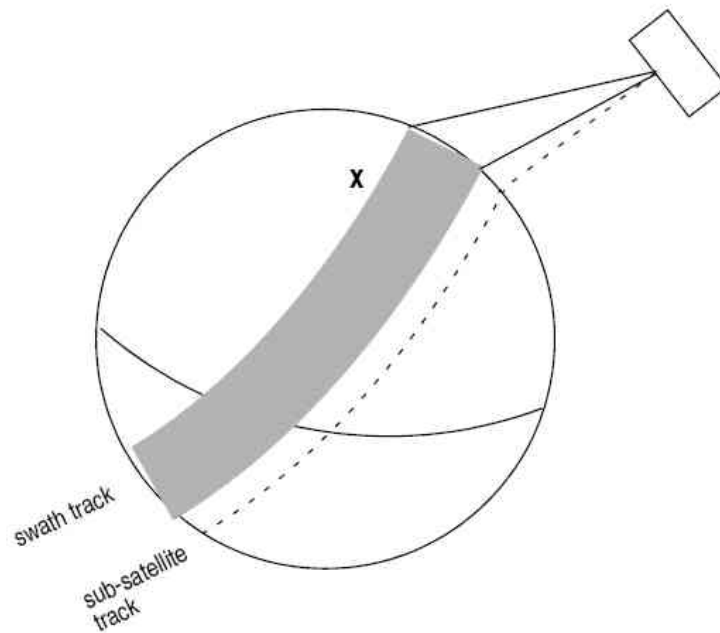
- instantaneous swath: the part of the earth surface observed by an instrument at a given time
- swath track: represents the track made on the earth surface by the instantaneous swath over a period of time

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

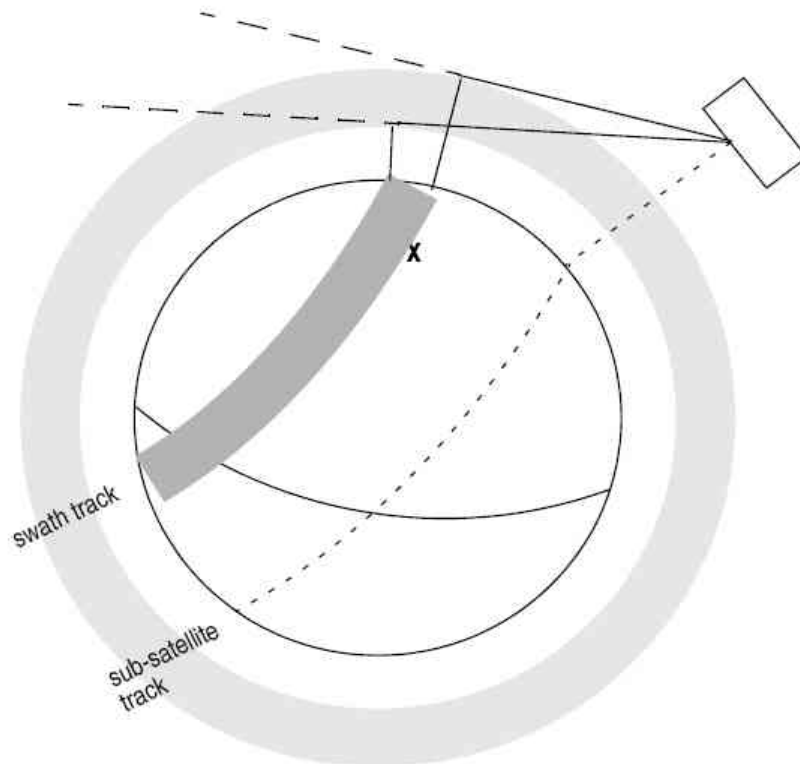
For limb sounding instruments, the concept can be generalized to define a “thick swath”. This is obtained by defining a minimum and a maximum altitude, and considering the tangent points to these altitudes as the edges of the swath. Two cases have to be considered:

- 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.

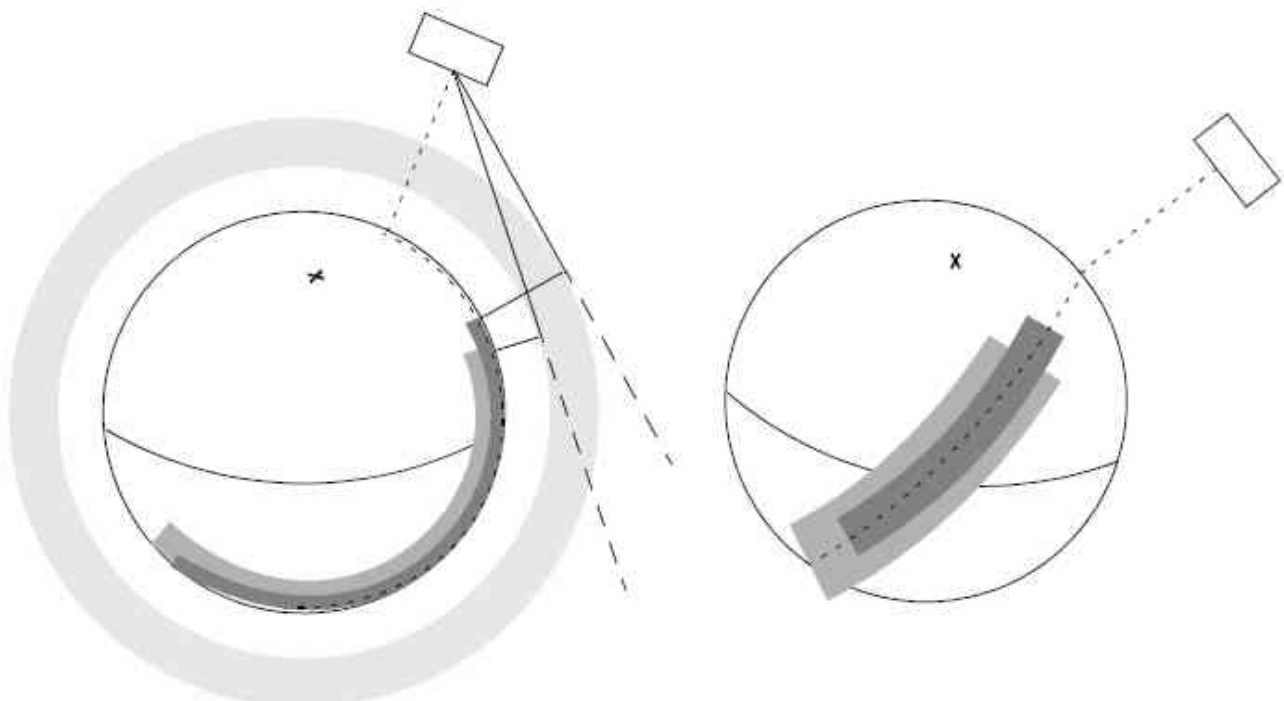
For these, 2 Instrument Swath Files are provided:

- 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 4: Envisat Swaths**

instrument	Mode	File Prefix = swath	Swath geometry (Table 56)	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 5);

- 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 5: 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).

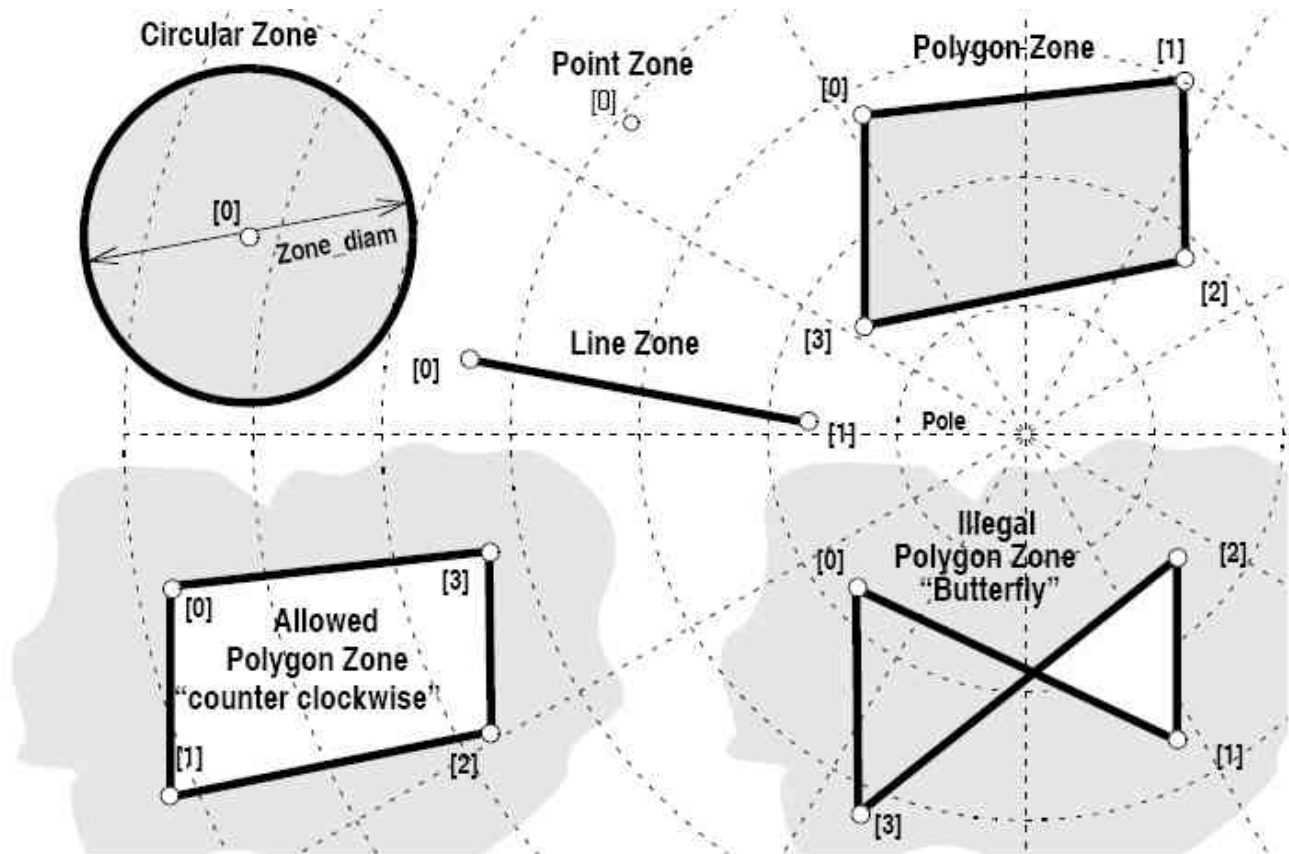


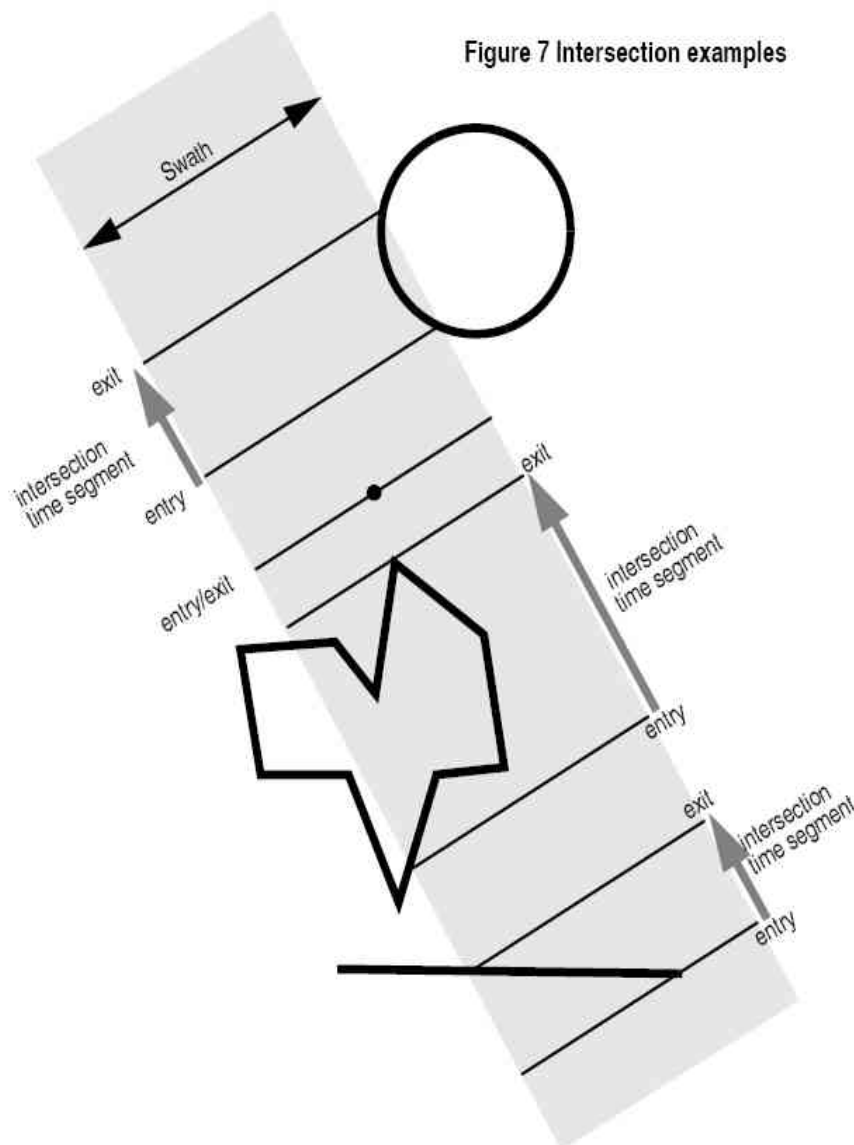
Figure 6: Zone examples

### 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  $\langle i \rangle$  to  $\langle j \rangle$  is defined in a clockwise manner inclusive point  $\langle i \rangle$  but exclusive point  $\langle j \rangle$ . The swath line from time  $\langle k \rangle$  to  $\langle l \rangle$  is defined inclusive the template point at  $\langle k \rangle$  but exclusive the template point at  $\langle l \rangle$ .

The fraction of the swath along track line determines the precise timing since time  $\langle k \rangle$  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

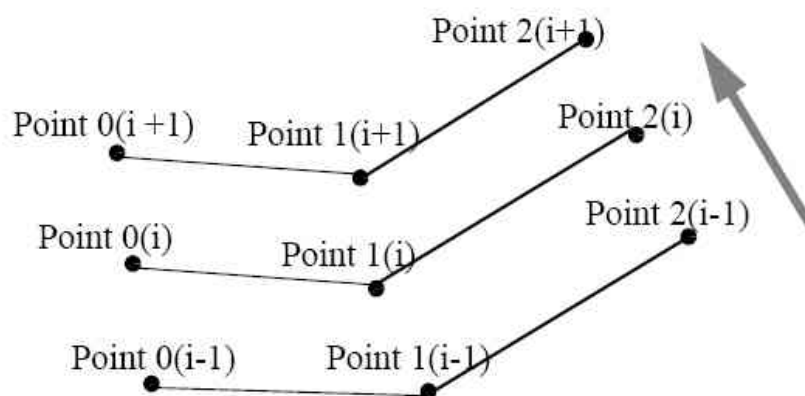
The left and right side of the swath, are located using the same algorithm as for the point swath. Even left and right time segments can be made based on the left and right hand transitions.

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  $\langle k \rangle$  to time  $\langle l \rangle$ , 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);

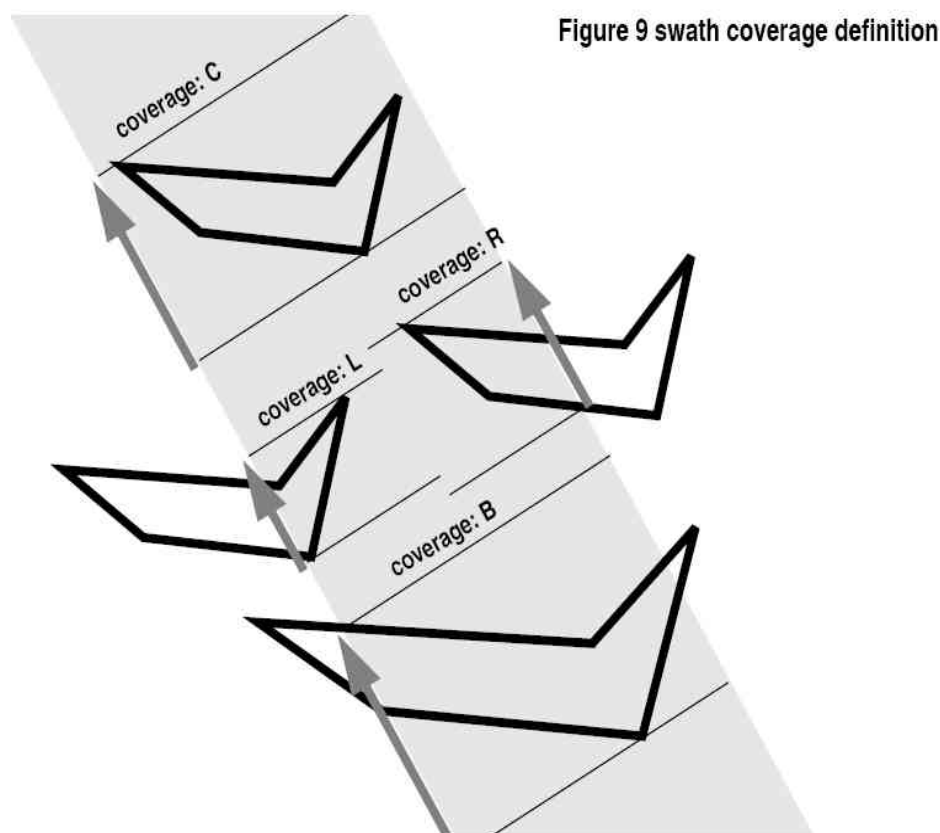


Figure 9: Swath coverage definition



- 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 6: Input parameters of `xv_zone_vis_time` function**

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>orbit_id</code>	<code>xo_orbit_id*</code>	-	Structure that contains the orbit data	-	-
<code>orbit_type</code>	<code>long*</code>	-	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 <code>orbit_id</code> was initialized with orbital changes (with <code>xo_orbit_init_def</code> or with <code>xo_orbit_init_file</code> plus an OSF file). In other cases, only the value <code>XV_ORBIT_ABS</code> can be used.	-	Complete.
<code>start_orbit</code>	<code>long</code>	-	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: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> <code>start_osf</code></li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
<code>start_cycle</code>	<code>long</code>	-	Cycle number corresponding to the <code>start_orbit</code> . Dummy when using absolute orbits	cycle number	= 0 or $\geq$ first cycle in osf
<code>stop_orbit</code>	<code>long</code>	-	Last orbit, segment filter.  For <code>orbit_id</code> initialized with orbital changes, when: <ul style="list-style-type: none"> <li><code>stop_orbit = 0</code> (for <code>orbit_type = XV_ORBIT_ABS</code>)</li> </ul> or <ul style="list-style-type: none"> <li><code>stop_orbit = 0</code> and <code>stop_cycle = 0</code> (for <code>orbit_type = XV_ORBIT_REL</code>)</li> </ul> the <code>stop_orbit</code> will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> <code>start_osf</code></li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>

			change of the start_orbit. <ul style="list-style-type: none"> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul>		
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: <ul style="list-style-type: none"> <li>0 = (XV_STF) if the swath file is a swath template file.</li> <li>&gt; 0 if the swath files is a swath definition file. In this case the swath points are generated for every "swath_flag" orbits</li> </ul>	-	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 <ul style="list-style-type: none"> <li>- circle centre, for circ. zone, i = 1</li> <li>- point, for point zone, i = 1</li> <li>- line-end, for line zone, i = 1 or 2</li> </ul>		

			- 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 7: Output parameters of `xv_zone_vis_time` function*

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>xv_zone_vis_time</code>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<code>number_segments</code>	long		Number of visibility segments returned to the user.		$\geq 0$
<code>bgn_orbit</code>	long*	all	Orbit number, begin of visibility segment i <code>bgn_orbit[i-1]</code> , i = 1, <code>number_segments</code>		> 0
<code>bgn_second</code>	long*	all	Seconds since ascending node, begin of visibility segment i <code>bgn_second[i-1]</code> , i = 1, <code>number_segments</code>	s	$\geq 0$ < orbital period
<code>bgn_microsec</code>	long*	all	Micro seconds within second begin of visibility segment i <code>bgn_microsec[i-1]</code> , i = 1, <code>number_segments</code>	$\mu$ s	$\geq 0$ 999999
<code>bgn_cycle</code>	long*	all	Cycle number, begin of visibility segment i <code>bgn_orbit[i-1]</code> , i = 1, <code>number_segments</code>		>0 NULL when using absolute orbits
<code>end_orbit</code>	long*	all	Orbit number, end of visibility segment i <code>end_orbit[i-1]</code> , i = 1, <code>number_segments</code>		> 0
<code>end_second</code>	long*	all	Seconds since ascending node, end of visibility segment i <code>end_second[i-1]</code> , i = 1, <code>number_segments</code>	s	$\geq 0$ < 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_Z ONE_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 8: 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_ERR	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_ERR	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_TIME	11

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

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

ERR	Cannot (re)allocate memory for the segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_SEGMENTS_MEMORY_ERR	42
ERR	Too many time segments (more than MAX_ORBITS).	Computation not performed	XV_CFI_ZONE_VIS_TIME_MAX_ORBITS_ERR	43
ERR	Cannot allocate memory for the coverage.	Computation not performed	XV_CFI_ZONE_VIS_TIME_COVERAGE_MEMORY_ERR	44
WARN	Warning checking the visibility segments.	Computation performed. Message to inform the user.	XV_CFI_ZONE_VIS_TIME_CHECK_SEGMENTS_WARN	45
ERR	Error checking the visibility segments.	Computation not performed	XV_CFI_ZONE_VIS_TIME_CHECK_SEGMENTS_ERR	46
ERR	Error computing final segments for the POINT swath and POINT zone.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_TO_TIME_CALL_ERR	47
ERR	Wrong input Orbit Id. Unknown orbit initialization mode	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_MODEL_ERR	48
WARN	"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_WARN	49
ERR	Error computing the ANX longitude	Computation not performed	XV_CFI_ZONE_VIS_TIME_COMPUTE_ANX_ERR	50
ERR	Error calling "orbit info"	Computation not performed	XV_CFI_ZONE_VIS_TIME_ORBIT_INFO_ERR	51
ERR	Error computing Multi- Point swath visibilities	Computation not performed	XV_CFI_ZONE_VIS_TIME_MULTI_POINT_SWATH_INTERSECTIONS_ERR	52
ERR	Error computing Point swath visibilities	Computation not performed	XV_CFI_ZONE_VIS_TIME_POINT_SWATH_INTERSECTIONS_ERR	53
ERR	Error checking visibility segments	Computation not performed	XV_CFI_ZONE_VIS_TIME_ON_OFF_CHECKING_ERR	54
ERR	Error merging visibility segments	Computation not performed	XV_CFI_ZONE_VIS_TIME_MERGE_SWATH_SEGMENTS_VISIBILITIES_ERR	55
ERR	Error trying to allocate memory	Computation not performed	XV_CFI_ZONE_VIS_TIME_MEMORY_ALLOCATION_ERR	56
ERR	Error calling "swath_pos"	Computation not performed	XV_CFI_ZONE_VIS_TIME	57

			ME_SWATH_POS_ER R	
ERR	Error calling "Polygon_inner_point"	Computation not performed	XV_CFI_ZONE_VIS_TI ME_POLYGON_INNE R_POINT_ERR	58
ERR	Error comparing orbits orbital changes	Computation not performed	XV_CFI_ZONE_VIS_TI ME_CHECK_ORBITAL_CH ANGE_ERR	59
ERR	Error converting zone point arrays to zone record	Computation not performed	XV_CFI_ZONE_VIS_TI ME_CONVERT_ZONE_ER R	64
ERR	No suitable zone found for orbit interval	Computation not performed	XV_CFI_ZONE_VIS_TI ME_ZONE_ORBIT_ERR	65
ERR	Error cloning zone	Computation not performed	XV_CFI_ZONE_VIS_TI ME_CLONE_ZONE_ERR	66
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_ZONE_VIS_TIME_ ORBIT_INTERVAL_STF_E RR	67
WARN	Input orbit interval is partially outside STF validity interval	Computation performed	XV_CFI_ZONE_VIS_TIME_ ORBIT_INTERVAL_STF_W ARN	68
ERR	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation not performed	XV_CFI_ZONE_VIS_TIME_ OSF_NON_LIN_STF_OLD_ WARN	69
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation performed	XV_CFI_ZONE_VIS_TIME_ SWATH_FLAG_LARGER_T HAN_LIN_APPROX_VAL_ WARN	70
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_ZONE_VIS_TIME_ GEO_SAT_ERR	71

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

## 7.2 xv\_zone\_vis\_time\_no\_file

### 7.2.1 Overview

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
<code>orbit_id</code>	<code>xo_orbit_id*</code>	-	Structure that contains the orbit data	-	-
<code>orbit_type</code>	<code>long*</code>	-	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 <code>orbit_id</code> was initialized with orbital changes (with <code>xo_orbit_init_def</code> or with <code>xo_orbit_init_file</code> plus an OSF file). In other cases, only the value <code>XV_ORBIT_ABS</code> can be used.	-	Complete.
<code>start_orbit</code>	<code>long</code>	-	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: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq \text{start\_osf}</math></li> <li>relative orbits <math>\leq \text{repeat cycle}</math></li> </ul>
• <code>start_cycle</code>	<code>long</code>	-	Cycle number corresponding to the <code>start_orbit</code> . Dummy when using absolute orbits	cycle number	= 0 or $\geq \text{first cycle in osf}$
<code>stop_orbit</code>	<code>long</code>	-	Last orbit, segment filter.  For <code>orbit_id</code> initialized with orbital changes, when: <ul style="list-style-type: none"> <li><code>stop_orbit = 0</code> (for <code>orbit_type = XV_ORBIT_ABS</code>)</li> </ul> or <ul style="list-style-type: none"> <li><code>stop_orbit = 0</code> and <code>stop_cycle = 0</code> (for <code>orbit_type = XV_ORBIT_REL</code>)</li> </ul> the <code>stop_orbit</code> will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq \text{start\_osf}</math></li> <li>relative orbits <math>\leq \text{repeat cycle}</math></li> </ul>



			<p>change of the start_orbit.</p> <ul style="list-style-type: none"> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul> <p>If it is not initialized with orbital changes, stop orbit will be set to the last orbit in orbit_id initialization.</p>		
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using absolute orbits	cycle number	= 0 or ≥ first cycle in osf
sff_data	xd_stf_file	-	<p>Swath template data (structure described in [D_H_SUM]).</p> <p>The swath structure can be got by:</p> <ul style="list-style-type: none"> <li>Reading a swath template file with the CFI function <b>xd_read_stf</b>.</li> <li>Generating the swath data with the CFI function <b>xv_gen_swath_no_file</b></li> </ul>	-	-
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 <b>xd_read_zone</b> .	-	-
projection	long		<p>projection used to define polygon sides as straight lines:</p> <p>= 0 Read projection from Zones DB</p> <p>= 1 Azimuthal gnomonic</p> <p>= 2 Rectangular lat/long</p>	-	-
zone_num	long		<p>Number of vertices of the zone provided in zone_long, zone_lat:</p> <p>= 0 no vertices provided, use zone_id / zone_db_file</p> <p>= 1 Point / Circular zone,</p> <p>= 2 Line zone</p> <p>&gt; 2 Polygon zone</p>		≥ 0
zone_long	double*	all	<p>zone_long[i-1]</p> <p>Geocentric longitude of</p> <p>- circle centre, for circ. zone, i =1</p> <p>- point, for point zone, i = 1</p>		

			- 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
<code>xv_zone_vis_time_no_file</code>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<code>number_segments</code>	long		Number of visibility segments returned to the user.		$\geq 0$
<code>bgn_orbit</code>	long*	all	Orbit number, begin of visibility segment <i>i</i> <code>bgn_orbit[i-1]</code> , <i>i</i> = 1, <code>number_segments</code>		> 0
<code>bgn_second</code>	long*	all	Seconds since ascending node, begin of visibility segment <i>i</i> <code>bgn_second[i-1]</code> , <i>i</i> = 1, <code>number_segments</code>	s	$\geq 0$ < orbital period
<code>bgn_microsec</code>	long*	all	Micro seconds within second begin of visibility segment <i>i</i> <code>bgn_microsec[i-1]</code> , <i>i</i> = 1, <code>number_segments</code>	$\mu$ s	$\geq 0$ 999999
<code>bgn_cycle</code>	long*	all	Cycle number, begin of visibility segment <i>i</i> <code>bgn_orbit[i-1]</code> , <i>i</i> = 1, <code>number_segments</code>		>0 NULL when using absolute orbits
<code>end_orbit</code>	long*	all	Orbit number, end of visibility segment <i>i</i> <code>end_orbit[i-1]</code> , <i>i</i> = 1, <code>number_segments</code>		> 0
<code>end_second</code>	long*	all	Seconds since ascending node, end of visibility segment <i>i</i> <code>end_second[i-1]</code> , <i>i</i> = 1, <code>number_segments</code>	s	$\geq 0$ < 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_Z ONE_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 8).

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).

## 7.3 **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\_Vailability\_Range* and *Stop\_Vailability\_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 66). 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 11: 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: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math>start_osf</li> <li>relative orbits <math>\leq</math>repeat cyc</li> </ul>
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> the stop_orbit will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul> 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: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math>start_osf</li> <li>relative orbits <math>\leq</math>repeat cycle</li> </ul>
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf

swath_flag	long*	-	Define the use of the swath file: <ul style="list-style-type: none"> <li>• 0 = (XV_STF) if the swath file is a swath template file.</li> <li>• &gt; 0 if the swath files is a swath definition file. In this case the swath points are generated for every "swath_flag" orbits</li> </ul>	-	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 12: Output parameters of *xv\_station\_vis\_time* function

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_station_vis_time</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>number_segments</i>	long		Number of visibility segments returned to the user		≥ 0
<i>bgn_orbit</i>	long*	all	Orbit number, begin of visibility segment <i>i</i> <i>bgn_orbit</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>		> 0
<i>bgn_second</i>	long*	all	Seconds since ascending node, begin of visibility segment <i>i</i> <i>bgn_second</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>	s	≥ 0 < orbital period
<i>bgn_microsec</i>	long*	all	Micro seconds within second begin of visibility segment <i>i</i> <i>bgn_microsec</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>	μs	≥ 0 ≤ 999999
<i>bgn_cycle</i>	long*	all	Cycle number, begin of visibility segment <i>i</i> <i>bgn_cycle</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>		> 0 NULL when using absolute orbits
<i>end_orbit</i>	long*	all	Orbit number, end of visibility segment <i>i</i> <i>end_orbit</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>		> 0
<i>end_second</i>	long*	all	Seconds since ascending node, end of visibility segment <i>i</i> <i>end_second</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>	s	≥ 0 < orbital period
<i>end_microsec</i>	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 13: 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_ERR	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_VI S_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_VI S_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_VI S_TIME_LAST_ORBIT_WARN	12
ERR	Actual stop orbit is earlier than actual start orbit.	Computation not performed	XV_CFI_STATION_VI S_TIME_WRONG_INTERVAL_ERR	13
ERR	Error obtaining orbital information in orbit info.	Computation not performed	XV_CFI_STATION_VI S_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_VI S_TIME_ORBIT_CHANGE_WARN	15
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_STATION_VI S_TIME_INTERNAL_MEMORY_ERR	16
ERR	There is a potential memory overload, try with a smaller orbital interval.	Computation not performed	XV_CFI_STATION_VI S_TIME_POTENTIAL_MEMORY_ERR	17
ERR	Orbital information does not coincide with reference swath.	Computation not performed	XV_CFI_STATION_VI S_TIME_INCONSISTENT_SWATH_ERR	18
ERR	Error read info the ground station's mask data file.	Computation not performed	XV_CFI_STATION_VI S_TIME_READ_STATION_ERR	19
ERR	Error transforming the station's mask into an equivalent zone.	Computation not performed	XV_CFI_STATION_VI S_TIME_AZEL2LONLAT_ERR	20
ERR	Error calling ZONEVISTIME to calculate transitions.	Computation not performed	XV_CFI_STATION_VI S_TIME_ZONE_VISTIME_CALL_ERR	21
ERR	Error refining intersection time.	Computation not performed	XV_CFI_STATION_VI S_TIME_CALL_STAVISS_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_VI S_TIME_CALL_STAVISS_WARN	23
ERR	Error allocating memory for the time segments.	Computation not performed.	XV_CFI_STATION_VI S_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_SEGMENT_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 in the mask type read from the mask data given in the file	Computation not performed	XV_CFI_STATION_VIS_TIME_MASK_FROM_FILE_MASK_TYPE_ERR	32
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	33
ERR	Error converting zone point array to zone record	Computation not performed	XV_CFI_STATION_VIS_TIME_CONVERT_ZONE_ERR	34
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_STATION_VIS_TIME_ORBIT_INTERVAL_STF_ERR	35
WARN	Input orbit interval is partially outside STF validity interval	Computation performed	XV_CFI_STATION_VIS_TIME_ORBIT_INTERVAL_STF_WARN	36
ERR	Input OSF has non-trivial MLST non linear terms but STF was generated without them	Computation not performed	XV_CFI_STATION_VIS_TIME_OSF_NON_LIN_STF_OLD_WARN	37
WARN	Swath flag larger than MLST linear approximation validity. MLST linear approximation validity used.	Computation performed	XV_CFI_STATION_VIS_TIME_SWATH_FLAG_LARGER_THAN_LIN_APPROX_VALID_WARN	38
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_STATION_VIS_TIME_GEO_SAT_ERR	39

## 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 14: 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: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> the stop_orbit will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul> 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: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ 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: <ul style="list-style-type: none"> <li>• Reading a swath template file with the CFI function <b>xd_read_stf</b>.</li> <li>• Generating the swath data with the CFI function <b>xv_gen_swath_no_file</b></li> </ul>		
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 <b>xd_read_station</b> .	-	-
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 15: Output parameters of `xv_station_vis_time_no_file` function

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

			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 13) .

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).

## 7.5 xv\_sc\_vis\_time

### 7.5.1 Overview

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

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

**xv\_sc\_vis\_time** requires access to the orbit\_id (xo\_orbit\_id) data structure of both satellites. For orbit id initialization please refer to [ORBIT\_SUM]. The time intervals used by **xv\_sc\_vis\_time** are expressed in absolute or relative orbit numbers (with respect to source satellite). This is valid for both:

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

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

Users who need to use processing times must make use of the conversion routines provided in [ORBIT\_SUM] (**xo\_time\_to\_orbit** and **xv\_orbit\_to\_time** functions).

The **xv\_sc\_vis\_time** function considers the following sources of occultation, which can be configured through xv\_link\_data input struct:

- Earth plus a minimum tangent height.
- Satellite inclusive and exclusive masks, which are zones of azimuth and elevation where visibility is possible (inclusive mask) or not possible (exclusive mask). With this masks all the restrictions (mechanical or of any type) must be simulated.

Notes about definition of masks:

- The masks are defined as closed zones.
- These zones are defined in the input struct xv\_link\_data with arrays of azimuth and elevation points that define a polygon in the azimuth-elevation plane (last point in array is closed with first point in array internally).
- It must be distinguished between azimuth = 0. deg and azimuth = 360. deg, since they are considered different in the azimuth-elevation plane; this has been done to make the definition of masks easier.

### 7.5.2 Calling interface

For C programs, the call to **xv\_sc\_vis\_time** is (input parameters are underlined):

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

```
xo_orbit_id orbit_id2 = {NULL};
xp_sat_nom_trans_id sat_nom_trans_id1 = {NULL};
xp_sat_nom_trans_id sat_nom_trans_id2 = {NULL};
xp_sat_trans_id      sat_trans_id1 = {NULL};
xp_sat_trans_id      sat_trans_id2 = {NULL};
xp_instr_trans_id    instr_trans_id1 = {NULL};
xp_instr_trans_id    instr_trans_id2 = {NULL};
long                 orbit_type,
                    start_orbit, start_cycle,
                    stop_orbit, stop_cycle,
                    number_segments,
                    *bgn_orbit, *bgn_second,
                    *bgn_microsec, *bgn_cycle,
                    *end_orbit, *end_second,
                    *end_microsec, *end_cycle,
                    ierr[XV_NUM_ERR_SC_VIS_TIME],
                    status;

double               min_duration;
xv_link_data         link_data;

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

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

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

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

### 7.5.3 Input parameters

Table 16: Input parameters of xv\_sc\_vis\_time

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id1	xo_orbit_id*	-	Structure that contains the orbit data of the source satellite	-	-
sat_nom_trans_id1	xp_sat_nom_trans_id*	-	Structure that contains the Sat. Nom. Trans. of the source satellite	-	-
sat_trans_id1	xp_sat_trans_id*	-	Structure that contains the Sat. Trans. of the source satellite	-	-
instr_trans_id1	xp_instr_trans_id*	-	Structure that contains the Instr. Trans. of the source satellite	-	-
orbit_type	long	-	Define the type of orbit representation, i.e. absolute or relative orbits in the input/output parameters	-	Complete
start_orbit	long	-	First orbit, segment filter. Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file) First Orbit in the orbit_scenario_file will be used when: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\leq$ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> the stop_orbit will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>

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

## 7.5.4 Output parameters

Table 17: Output parameters of *xv\_sc\_vis\_time* function

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

			end of visibility segment i end_microsec[i-1], i = 1, number_segments		≤ 999999
end_cycle	long*	all	Cycle number, begin of visibility segment i bgn_cycle[i-1], i = 1, number_segments		>0 NULL when using absolute orbits
ierr[XV_NUM_E RR_SC_VIS_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\_sc\_vis\_time** function. So the user will only have to declare those pointers but not to allocate memory for them. However, once the function has returned without error, the user will have the responsibility of freeing the memory for those pointers once they are not used.



## 7.5.5 Warnings and errors

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

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

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

**Table 18: Error messages of `xv_sc_vis_time`**

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

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

## 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<sup>1</sup>:

- longitude
- latitude
- altitude

for  $n$  points (with  $n \geq 1$ ). 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 and no computation will be performed.

---

<sup>1</sup> 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 19: 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 20: 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 21: 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_ERR	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_ERR	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_ERR	14
ERR	Error while transforming from relative to absolute orbit.	Computation not performed	XV_CFI_SWATH_POS_REL_TO_ABS_ERR	15
ERR	Error while computing information of the orbit.	Computation not performed	XV_CFI_SWATH_POS_XV_ORBIT_INFO_ERR	16
ERR	The swath template structure contains invalid data	Computation not performed	XV_CFI_SWATH_POS_SWATH_INIT_ERR	17
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_SWATH_POS_MEMORY_ERR	18
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_SWATH_POS_GEO_SAT_ERR	19



## 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).

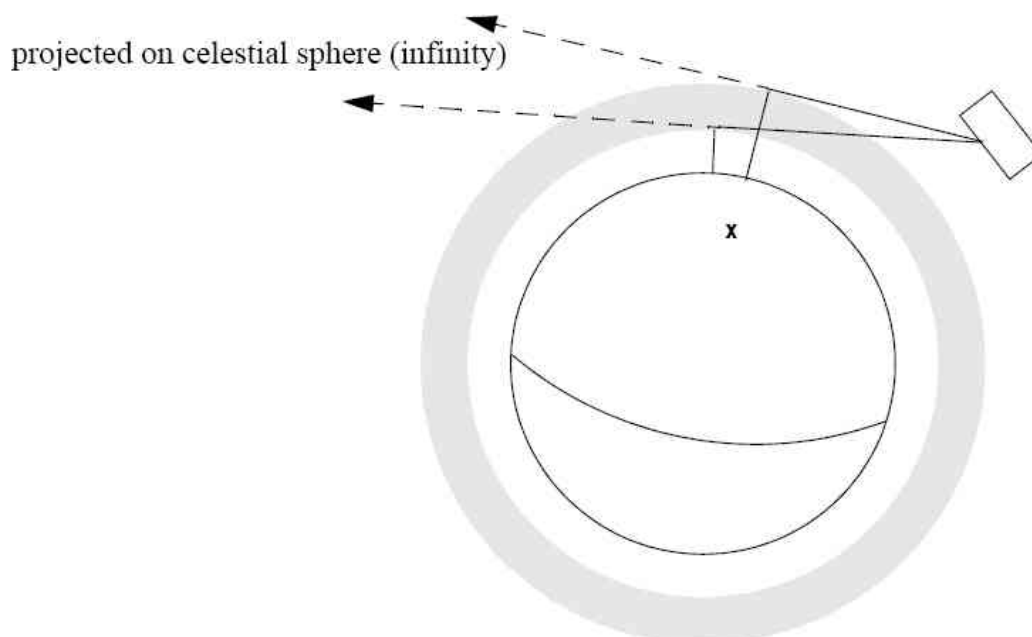
## 7.7.2 Swath Definition

`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



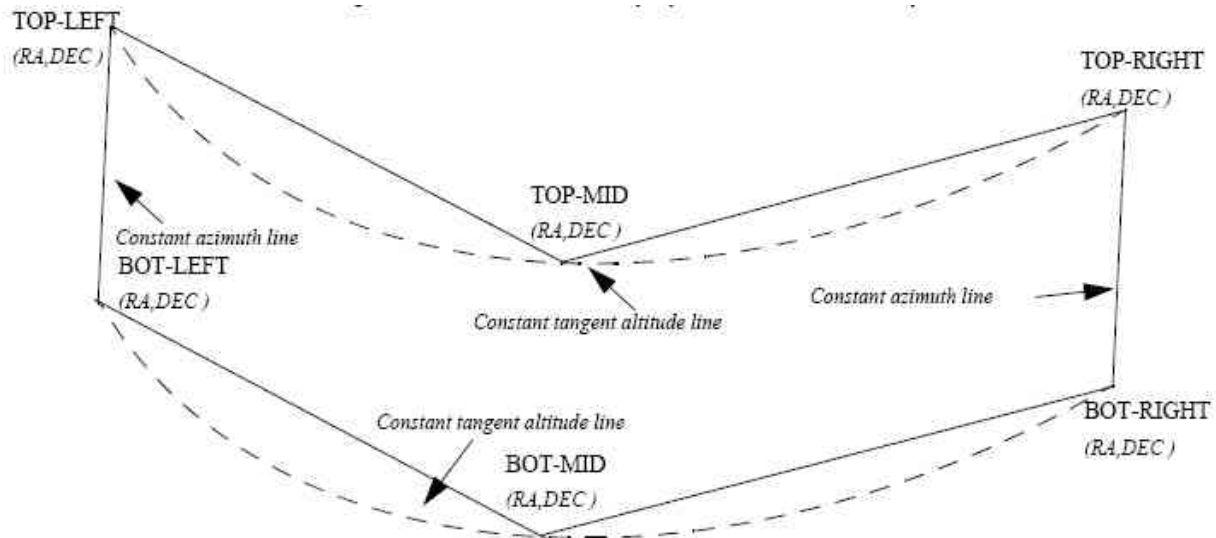
**Figure 10: Two tangent altitudes over the ellipsoid**

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_star_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 103). 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 22: 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 $\geq$ start_osf relative orbits $\leq$ repeat cycle
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ start_osf
stop_orbit	long	-	Last orbit, segment filter. When: <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> the stop_orbit will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul> 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 $\geq$ start_osf relative orbits $\leq$ repeat cycle
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	=0 or $\geq$ start_osf
swath_flag	long*	-	Define the use of the swath file: <ul style="list-style-type: none"> <li>0 = (XV_STF) if the swath file is a</li> </ul>	-	XV_STF = 0 XV_SDF = 1

			swath template file. <ul style="list-style-type: none"> <li>&gt; 0 if the swath files is a swath definition file. In this case the swath points are generated for every "swath_flag" orbits</li> </ul>		> 0
swath_file_upper	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_lower	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 <b>ONLY IF</b> 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 <b>ONLY IF</b> star_db_file is equal empty string(“”)	deg	(-180.0, 180.0)
star_dec	double*		Declination of Star, in TOD.  This parameter is used <b>ONLY IF</b> 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 23: Output Parameters of *xv\_star\_vis\_time*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_star_vis_time</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>star_ra_deg</i>	double		Right Ascension of the star, in TOD, for the UTC halfway <i>start_orbit</i> and <i>stop_orbit</i> .	deg	(-180.0, 180.0)
<i>star_dec_deg</i>	double		Declination of the star, in TOD, for the UTC halfway <i>start_orbit</i> and <i>stop_orbit</i> .	deg	(-90.0, 90.0)
<i>number_segment</i>	long		Number of visibility segments returned to the user		≥ 0
<i>bgn_orbit</i>	long*	all	Orbit number, begin of visibility segment <i>i</i> <i>bgn_orbit</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>		> 0
<i>bgn_second</i>	long*	all	Seconds since ascending node, begin of visibility segment <i>i</i> <i>bgn_second</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>	s	≥ 0 < orbital period
<i>bgn_microsec</i>	long*	all	Micro seconds within second begin of visibility segment <i>i</i> <i>bgn_microsec</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>	μs	≥ 0 ≤ 999999
<i>bgn_cycle</i>	long*	all	cycle number begin of visibility segment <i>i</i> <i>bgn_microsec</i> [ <i>i</i> -1], <i>i</i> = 1, <i>number_segments</i>		> 0 NULL when using relative orbits
<i>bgn_coverage</i>	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 24: 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
WARN	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
WARN	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
WARN	Warning, there is an orbital change within the requested orbits.	Computation performed Message to inform the user	XV_CFI_STAR_VIS_TIME_ORBIT_CHANGE_WARN	3
ERR	Error, starvistime can only operate with an inertial swath.	Computation not performed	XV_CFI_STAR_VIS_TIME_INERTIAL_SWATH_ERR	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_TIME	10

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

	in range	performed	ORBITAL_CHANGE_ERR	
ERR	Input orbit interval is completely outside STF validity interval	Computation not performed	XV_CFI_STAR_VIS_TIME_ORBIT_INTERVAL_STF_ERR	27
WARN	Input orbit interval is partially outside STF validity interval	Computation performed	XV_CFI_STAR_VIS_TIME_ORBIT_INTERVAL_STF_WARN	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

## 7.8 **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).

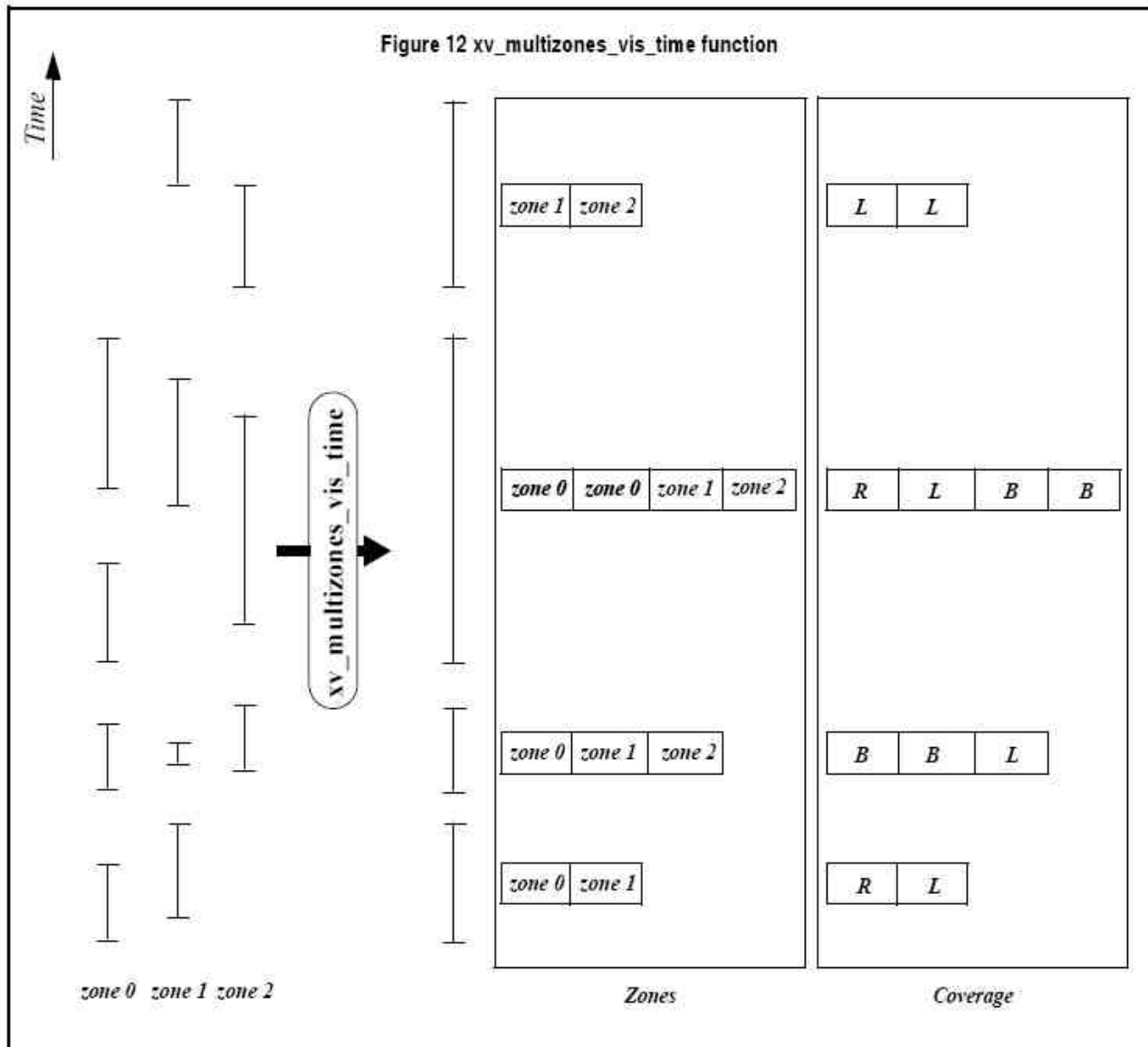


Figure 12: xv\_multizones\_vis\_time function

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 25: 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: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
• start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter. When: <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> the stop_orbit will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
swath_flag	long*	-	Define the use of the swath file: <ul style="list-style-type: none"> <li>0 = (XV_STF) if the swath file is a swath template file.</li> <li>&gt; 0 if the swath files is a swath</li> </ul>	-	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 <sup>2</sup> .	deg	
zone_lat	double*	all	Geodetic latitude of  - circle centre, for circ. 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]
- ...

			<ul style="list-style-type: none"> <li>- point, for point zone.</li> <li>- line-end, for line zone.</li> <li>- vertices, for polygon zone.</li> </ul> <p>The latitude of the vertices corresponding to all zones shall be arranged consecutively.</p>		
zone_diam	double*	all	<p>Array of diameters of circular zones in case this shape is selected for any zone<sup>3</sup>.</p> <p>zone_diam=0.0 for Point Zones.</p>	m	≥ 0.0
min_duration	double	-	<p>Minimum duration for segments.</p> <p>Only segments with a duration longer than min_duration will be given on output.</p>	s	≥ 0
extra_info_flag	long	-	<p>If value set to false (= 0), the zones_in_segment and coverage arrays are not computed.</p> <p>Saves computation time.</p>		0 (false), 1 (true)

<sup>3</sup> 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 26: Output parameters of *xv\_multizones\_vis\_time*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_multizones_vis_time</i>	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 27: 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_MULTIZONE S_VIS_TIME_MEMOR Y_ERR	0
ERR	Error getting visibility segments for zone %ld	Computation not performed	XV_CFI_MULTIZONE S_VIS_TIME_COMPUT E_SEGMENTS_ERR	1
ERR	Error getting absolute orbit from relative orbit	Computation not performed	XV_CFI_MULTIZONE S_VIS_TIME_ABS_TO _REL_ORBIT_ERR	2
ERR	Error getting relative orbit vector from absolute orbits	Computation not performed	XV_CFI_MULTIZONE S_VIS_TIME_ABS_TO _REL_VECTOR_ERR	3
ERR	Error while merging overlapped segments	Computation not performed	XV_CFI_MULTIZONE S_VIS_TIME_OVERLA P_ERR	4
ERR	Orbit id is not initialized.	Computation not performed	XV_CFI_MULTIZONES_VIS _TIME_ORBIT_STATUS_E RR	5
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_MULTIZONES_VIS _TIME_GEO_SAT_ERR	6



## 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 28: 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: <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
• start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
stop_orbit	long	-	Last orbit, segment filter.  When: <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> the stop_orbit will be set to the minimum value between: <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e. the input orbit range will be a complete cycle)</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
stop_cycle	long	-	Cycle number corresponding to the stop_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
swath_flag	long*	-	Define the use of the swath file: <ul style="list-style-type: none"> <li>0 = (XV_STF) if the swath file is a swath template file.</li> <li>&gt; 0 if the swath files is a swath</li> </ul>	-	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 29: Output parameters of *xv\_multistations\_vis\_time*

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

z dop_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
z dop_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 z dop\_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 30: 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_ERR	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_ERR	3
ERR	Error while merging overlapped segments.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_OVERLAP_ERR	4
ERR	Orbit id not initialized.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_ORBIT_STATUS_ERR	5
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_MULTISTATIONS_VIS_TIME_GEO_SAT_ERR	6

## 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 zenith angles are reached (both up and down times are computed)
- Sun occultations by the Earth.
- Sun occultations 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 31: 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_IN FO_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 32: Output parameters of *xv\_orbi\_extra*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_orbit_extra</i>	long	-	Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>sza_up</i>	double	all	Seconds since ANX of Sun Zenith Angles when SZA is increasing with time.	s	≥ 0 ≤ orb. period
<i>sza_down</i>	double	all	Seconds since ANX of Sun Zenith Angles when SZA is decreasing with time.	s	≥ 0 ≤ orb. period
<i>eclipse_entry</i>	double	-	Seconds since ANX of eclipse entry.  Note that the value is provided within the input orbit, so that the <i>eclipse_exit</i> will be less than the <i>eclipse_entry</i> if the ANX is in eclipse.	s	≥ 0 ≤ orbital period -1 if there is not eclipse
<i>eclipse_exit</i>	double	-	Seconds since ANX of eclipse exit. Note that the value is provided within the input orbit, so that the <i>eclipse_exit</i> will be less than the <i>eclipse_entry</i> if the ANX is in eclipse.	s	≥ 0 ≤ orbital period -1 if there is not eclipse
<i>sun_moon_entry</i>	double	-	Seconds since ANX of Sun Occultation by Moon entry.	s	<-1 if no occultation is found ≥ 0 ≤ orbital period
<i>sun_moon_exit</i>	double	-	Seconds since ANX of Sun Occultation by Moon exit	s	<-1 if no occultation is found ≥ 0 ≤ orbital period
<i>ierr</i>	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 33: 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_EXTR A_ORBIT_STATUS_ER R	0
ERR	Error allocating memory for SZA entry/exit times	Computation not performed	XV_CFI_ORBIT_EXTR A_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_EXTR A_ECLIPSE_ERR	3
ERR	Error computing Sun occultation by Moon.	Computation not performed	XV_CFI_ORBIT_EXTR A_SUN_OCC_BY_MO ON_ERR	4

## 7.11 xv\_gps\_vis\_time

TBW

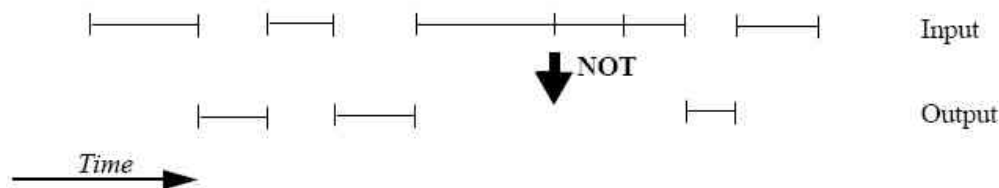


## 7.12 `xv_time_segments_not`

### 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 34: 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 35: Output parameters of *xv\_time\_segments\_not*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_time_segments_not</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>num_segments_out</i>	long	-	Number of segments in the output list.	-	>0
<i>bgn_orbit_out</i>	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
<i>bgn_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
<i>bgn_microsecs_out</i>	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 < 999999
<i>bgn_cycle_out</i>	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
<i>end_orbit_out</i>	long*	all	Array of orbit numbers for the end of the segments	-	>0
<i>end_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
<i>end_microsecs_out</i>	long*	all	Array of microseconds within a second for the end of the segments	-	>0 < 999999
<i>end_cycle_out</i>	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
<i>ierr[10]</i>	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 36: Error messages and codes**

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_REL_TO_ABS_ORBIT_ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_ABS_TO_REL_ORBIT_ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_NOT_GEO_SAT_ERR	4

## 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)

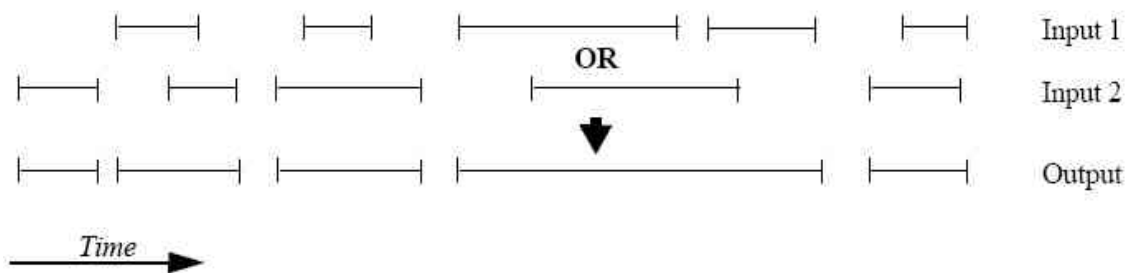


Figure 14: `xv_time_segments_or` function

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 37: 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 <i>XV_TIME_ORDER</i> 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 38: Output parameters of *xv\_time\_segments\_or*

C name	C type	Array Element	Description	Unit (Format )	Allowed Range
<i>xv_time_segments_or</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>num_segments_out</i>	long	-	Number of segments in the output list.	-	>0
<i>bgn_orbit_out</i>	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
<i>bgn_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
<i>bgn_microsecs_out</i>	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
<i>bgn_cycle_out</i>	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
<i>end_orbit_out</i>	long*	all	Array of orbit numbers for the end of the segments	-	>0
<i>end_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
<i>end_microsecs_out</i>	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
<i>end_cycle_out</i>	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
<i>ierr[10]</i>	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 39: Error messages and codes**

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

## 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)

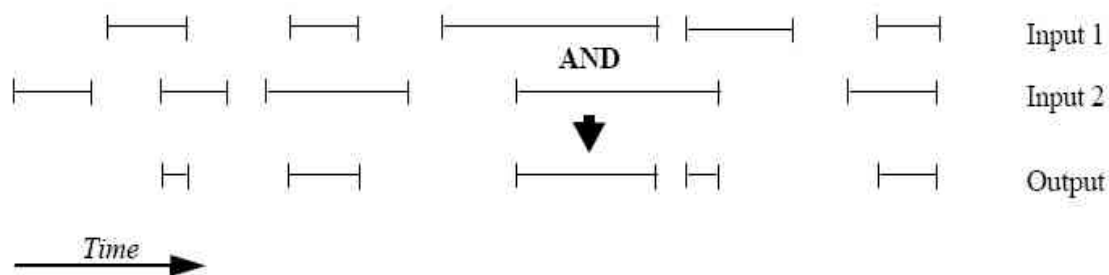


Figure 15: xv\_time\_segments\_and\_function

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 40: 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 41: Output parameters of *xv\_time\_segments\_and*

C name	C type	Array Element	Description	Unit (Format )	Allowed Range
<i>xv_time_segments_and</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>num_segments_out</i>	long	-	Number of segments in the output list.	-	>0
<i>bgn_orbit_out</i>	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
<i>bgn_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
<i>bgn_microsecs_out</i>	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
<i>bgn_cycle_out</i>	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
<i>end_orbit_out</i>	long*	all	Array of orbit numbers for the end of the segments	-	>0
<i>end_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
<i>end_microsecs_out</i>	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
<i>end_cycle_out</i>	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
<i>ierr[10]</i>	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 42: Error messages and codes**

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

## 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 43) depending on the orbit representation and the sort criteria chosen:

**Table 43: 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 44: 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 45: Output parameters of *xv\_time\_segments\_sort*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_time_segments_and</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>ierr</i> [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 46: Error messages and codes**

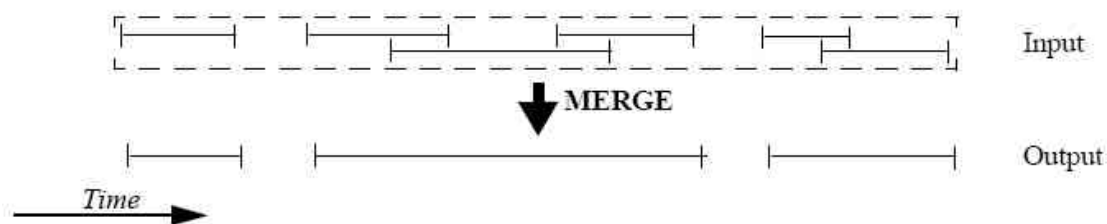
Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error allocating internal memory.	Computation not performed	XV_CFI_TIME_SEGMENTS_SORT_MEMORY_ERR	0
ERR	Error getting absolute orbit vector from relative orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_SORT_CHANGING_ORBIT_ERR	1
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_SORT_GEO_SAT_ERR	2

## 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 47: 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 48: Output parameters of *xv\_time\_segments\_merge*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_time_segments_merge</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>num_segments_out</i>	long	-	Number of segments in the output list.	-	>0
<i>bgn_orbit_out</i>	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
<i>bgn_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
<i>bgn_microsecs_out</i>	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
<i>bgn_cycle_out</i>	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
<i>end_orbit_out</i>	long*	all	Array of orbit numbers for the end of the segments	-	>0
<i>end_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
<i>end_microsecs_out</i>	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
<i>end_cycle_out</i>	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
<i>ierr[10]</i>	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 49: 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_ORBIT_ERR	1
ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_MERGE_ABS_TO_REL_ORBIT_ERR	2
ERR	Error sorting input list.	Computation not performed	XV_CFI_TIME_SEGMENTS_MERGE_SORTING_ERR	3
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MERGE_GEO_SAT_ERR	4

## 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 (input parameters are underlined):

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

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

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

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

### 7.17.3 Input parameters *xv\_time\_segments\_delta*

Table 50: 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 51: Output parameters of *xv\_time\_segments\_delta*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_time_segments_delta</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>num_segments_out</i>	long	-	Number of segments in the output list.	-	>0
<i>bgn_orbit_out</i>	long*	all	Array of orbit numbers for the beginning of the segments	-	>0
<i>bgn_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
<i>bgn_microsecs_out</i>	long*	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
<i>bgn_cycle_out</i>	long*	all	Array of cycle numbers for the beginning of the segments.	-	>0
<i>end_orbit_out</i>	long*	all	Array of orbit numbers for the end of the segments	-	>0
<i>end_secs_out</i>	long*	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
<i>end_microsecs_out</i>	long*	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
<i>end_cycle_out</i>	long*	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
<i>ierr[10]</i>	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 52: 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_ERR	3
ERR	Error transforming from processing times to orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_TIME_TO_ORBIT_ERR	4
ERR	Error modifying time segment duration	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_TIME_ADD_ERR	5
ERR	Error sorting input list	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_SORT_ERR	6
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_DELTA_GEO_SAT_ERR	7

## 7.18 xv\_time\_segments\_mapping

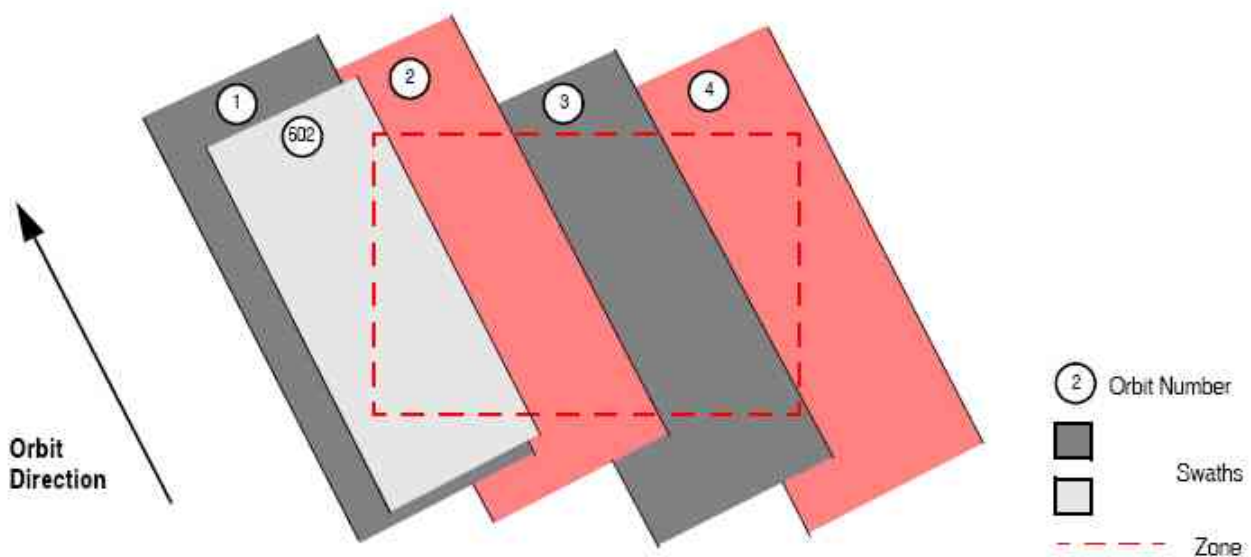
### 7.18.1 Overview

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 53: 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	-	<p>First orbit, segment filter</p> <p>Segments will be filtered as from the beginning of first orbit (within orbit range from orbit_scenario_file)</p> <p>First Orbit in the orbit_scenario_file will be used when:</p> <ul style="list-style-type: none"> <li>Absolute orbit is set to zero.</li> <li>Relative orbit and cycle number set to zero.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>
start_cycle	long	-	Cycle number corresponding to the start_orbit. Dummy when using relative orbits	cycle number	= 0 or $\geq$ first cycle in osf
stop_orbit	long	-	<p>Last orbit, segment filter.</p> <p>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.</p> <p>When:</p> <ul style="list-style-type: none"> <li>stop_orbit = 0 (for orbit_type = XV_ORBIT_ABS)</li> <li>stop_orbit = 0 and stop_cycle = 0 (for orbit_type = XV_ORBIT_REL)</li> </ul> <p>the stop_orbit will be set to the minimum value between:</p> <ul style="list-style-type: none"> <li>the last orbit within the orbital change of the start_orbit.</li> <li>start_orbit+cycle_length-1 (i.e.</li> </ul>	absolute or relative orbit number	= 0 or: <ul style="list-style-type: none"> <li>absolute orbits <math>\geq</math> start_osf</li> <li>relative orbits <math>\leq</math> repeat cycle</li> </ul>

			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: <ul style="list-style-type: none"> <li>• 0 = (XV_STF) if the swath file is a swath template file.</li> <li>• &gt; 0 if the swath files is a swath definition file. In this case the swath points are generated for every "swath_flag" orbits</li> </ul>	-	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		

			<ul style="list-style-type: none"> <li>- circle centre, for circ. zone, i =1</li> <li>- point, for point zone, i = 1</li> <li>- line-end, for line zone, i = 1 or 2</li> <li>- vertices, for polygon zone, i = 1... zone_num</li> </ul>		
zone_lat	double*	all	zone_lat[i-1] Geodetic latitude of <ul style="list-style-type: none"> <li>- circle centre, for circ. zone, i =1</li> <li>- point, for point zone, i = 1</li> <li>- line-end, for line zone, i = 1 or 2</li> <li>- vertices, for polygon zone, i = 1... zone_num</li> </ul>		

## 7.18.4 Output parameters *xv\_time\_segments\_mapping*

Table 54: Output parameters of *xv\_time\_segments\_mapping*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
<i>xv_time_segments_mapping</i>	long		Function status flag, = 0 No error > 0 Warnings, results generated < 0 Error, no results generated		
<i>num_mappings</i>	long		Number of output mappings		$\geq 0$
<i>num_segments</i>	long*	all	<i>num_segments</i> [n] = number of segments for the n-th mapping. n=0... ( <i>num_mappings</i> -1)	-	> 0
<i>orbit_direction</i>	long*	all	Direction of the segments of a mapping.	-	Complete (see Table 2: segment direction)
<i>bgn_orbit</i>	long**	all	Array of orbit numbers for the beginning of the segments	-	>0
<i>bgn_secs</i>	long**	all	Array of seconds elapsed since ANX for the beginning of the segments	-	>0 <nodal period
<i>bgn_microsecs</i>	long**	all	Array of microseconds within a second for the beginning of the segments	-	>0 <999999
<i>bgn_cycle</i>	long**	all	Array of cycle numbers for the beginning of the segments.	-	>0
<i>end_orbit</i>	long**	all	Array of orbit numbers for the end of the segments	-	>0
<i>end_secs</i>	long**	all	Array of seconds elapsed since ANX for the end of the segments	-	>0 <nodal period
<i>end_microsecs</i>	long**	all	Array of microseconds within a second for the end of the segments	-	>0 <999999
<i>end_cycle</i>	long**	all	Array of cycle numbers for the end of the segments.	-	>0 or NULL
<i>coverage</i>	long**	all	coverage of the output segments.	-	complete see Table 2
<i>ierr</i>	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 55: 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_ZONE_VIS_TIME_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_ERR	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_MEMORY_ERR	8
ERR	Error sorting segments.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SORT_ERR	9

ERR	Error getting relative orbit vector from absolute orbits.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_ABS_TO_REL_ERR	10
ERR	Error checking extremes of the orbit range.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_CHECK_EXTREMES_ERR	11
ERR	Error calling xv_swath_pos function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_POS_ERR	12
ERR	Error loading swath template file: %s	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_SWATH_READ_ERR	13
ERR	Swath file is not a line swath.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_INCORRECT_SWATH_ERR	14
WARN	Cannot check segments for start and stop orbits. Incomplete mappings could be generated.	Previous orbit to input start orbit and/or next orbit to the input stop orbit are not in the same orbital change that the input orbit range. It can not be checked whether there are segments missing at the extremes of the orbit range.  Computation performed.	XV_CFI_TIME_SEGMENTS_MAPPING_NO_CHECK_PERFORMED_WARN	15
WARN	Incomplete ascending mapping. %ld more track(s) needed to complete the mapping.	Computation performed.	XV_CFI_TIME_SEGMENTS_MAPPING_ASC_INCOMPLETE_MAPPING_WARN	16
WARN	Incomplete descending mapping. %ld more track(s) needed to complete the mapping.	Computation performed.	XV_CFI_TIME_SEGMENTS_MAPPING_DESC_INCOMPLETE_MAPPING_WARN	17
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_TIME_SEGMENTS_MAPPING_GEO_SAT_ERR	18



## 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 56). 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 ( $m \geq 1$ ). 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 56: Swath geometry definition (algorithm)**

Geometry ( <code>XD_Swath_geom_enum</code> )	Algorithm description	Swath point type ( <code>XD_Swath_point_type_enum</code> )
Pointing_Geometry (azimuth, elevation, altitude)	Swath point computed with <code>xp_target_inter</code> with that azimuth, elevation and altitude	Geodetic
Distance_Geometry (azimuth, elevation, altitude, distance)	Swath point computed with <code>xp_target_ground_range</code> with that azimuth, elevation, altitude and distance	Geodetic
Limb_Geometry (azimuth and altitude)	Swath point computed with <code>xp_target_altitude</code> with that azimuth and altitude	Geodetic
Inertial_Geometry (azimuth and altitude)	Swath point computed with <code>xp_target_altitude</code> 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 57: Input parameters of `xv_gen_swath` function*

C name	C type	Array Element	Description	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data.	-	-
atmos_id	xp_atmos_id*	-	Structure that contains the atmosphere initialisation.  This parameter determines the atmospheric and raytracing model used for the STF generation. The refraction model in the SDF is not used.	-	-
requested_orbit	long*	-	Orbit for which the instrument swath template file will be calculated.	absolute orbit number	> 0
swath_definition_file	char*	-	File name of the instrument swath definition file	-	-
dir_name	char*	-	Directory where the resulting STF is written (if empty (i.e. ""), the current directory is used)	-	-
swath_file	char*	-	Name for output swath file.  If empty (i.e. ""), the software will generate the name according to file name specification presented in [FORMATS], in this case the generated name is returned in this variable	-	-
file_class	char*	-	File class for output swath file	-	-
version_number	long*	-	Version number of output swath file	-	>= 1
fh_system	char*	-	System field of the output swath file fixed header	-	-

### 7.19.4 Output parameters

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

*Table 58: Output parameters of `xv_gen_swath` function*

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
<code>swath_file</code>	<code>char*</code>	-	Name for output swath file.  <u>This is only an output parameter when it is empty</u> (i.e. ""; see description of this parameter in Table 57)	-	-
<code>ierr[XV_ERR_VECTOR_MAX_LENGTH]</code>	<code>long</code>	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 59: Error messages of xv\_gen\_swath function**

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Error, wrong orbit Id.	Computation not performed	XV_CFI_GEN_SWATH_ORBIT_INIT_ERR	0
ERR	Wrong requested orbit	Computation not performed	XV_CFI_GEN_SWATH_REQUESTED_ORBIT_ERR	1
ERR	Could not get the creation date	Computation not performed	XV_CFI_GEN_SWATH_CURRENT_TIME_ERR	2
ERR	Error transforming time formats	Computation not performed	XV_CFI_GEN_SWATH_TIME_CONVERSION_ERR	3
ERR	Could not create the filename	Computation not performed	XV_CFI_GEN_SWATH_CREATE_FILENAME_ERR	4
ERR	Error reading swath definition file: %s	Computation not performed	XV_CFI_GEN_SWATH_SDF_READ_ERR	5
ERR	Error computing the swath points	Computation not performed	XV_CFI_GEN_SWATH_XV_ALGOR_ERR	6
ERR	Could not write the swath template file to disk	Computation not performed	XV_CFI_GEN_SWATH_WRITE_ERR	7
ERR	Wrong input file name. The file cannot be created	Computation not performed	XV_CFI_GEN_SWATH_WRONG_FILENAME_ERR	8
WARN	Could not find the input directory \"%s\". The current directory will be used instead	Computation performed	XV_CFI_GEN_SWATH_NO_DIR_WARN	9
ERR	Error allocating memory	Computation not performed	XV_CFI_GENSWATH_MEMORY_ERR	10
ERR	Geostationary satellite not allowed for this function.	Computation not performed	XV_CFI_GENSWATH_GEO_SAT_ERR	11

## 7.19.6 Executable Program

The `gen_swath` executable program can be called from a Unix shell as:

```
gen_swath  -sat satellite_name
           -sdf swath_definition_file_name
           -file orbit_file_name -orbit orbit_number
           [-tle]
           [-dir dir_name] (current directory by default)
           [-stf swath_template_filename] (empty string by default)
           [-precf file_name] (empty string by default)
           [-fcl 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:** The atmospheric model used for the STF generation is taken from the “Refraction” parameters in the SDF. The allowed values for the Refraction model in the input SDF are NO\_REF, STD\_REF or PRED\_REF. Note the user defined models are not allowed.

Example:

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

## 7.20 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 60: 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 <b>xd_read_sdf</b> .	-	-
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 61: 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 59).

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).

## 7.21 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 62: 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 containing 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 63: 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. <u>This is only an output parameter when it is empty</u> (i.e. “”; see description of this parameter in Table 62)	-	-
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 64: 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_ERR	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

## 8 RUNTIME PERFORMANCES

The library performance has been measured by dedicated test procedures run in 5 different platforms under the below specified machines:

<i>OS ID</i>	<i>Processor</i>	<i>OS</i>	<i>RAM</i>
LINUX64	Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz	GNU LINUX 2.6.24-16-generic (Ubuntu 8.04)	4 GB
LINUX32_LEGACY	Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz	GNU LINUX 2.6.24-16-generic (Ubuntu 8.04)	4 GB
LINUX64_LEGACY	Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz	GNU LINUX 2.6.24-16-generic (Ubuntu 8.04)	4 GB
MACIN64	Intel(R) Core(TM)2 Duo @ 2.00GHz	MAC OS X V10.5.8	1 GB
WINDOWS	Intel(R) Core(TM)2 Duo CPU T8100 @ 2.10 GHz	Microsoft Windows XP Professional Version 2002 Service Pack 3	2.99 GB

The table below shows the time (in milliseconds - ms) each function takes to be run under each platform:

<i>Function id</i>	<i>LINUX64</i>	<i>LINUX32 LEGACY</i>	<i>LINUX64 LEGACY</i>	<i>MACIN64</i>	<i>WINDOWS</i>
xv_swath_pos	0,051000	0,059000	0,072000	0,055000	0,131200
xv_zone_vis_time * 50 orbits	45,200001	43,599998	46,700001	73,599998	319,380005
xv_zone_vis_time_n o_file *	2,600000	4,600000	3,000000	4,400000	12,340000
50 orbits					
xv_time_segments_ not *	0,001700	0,002200	0,001800	0,003000	0,003590
34 segments					
xv_time_segments_ or *	0,003600	0,005800	0,003800	0,007500	0,008910
34 segments					

xv_time_segments_and*	0,008000	0,008300	0,005800	0,011400	0,013590
34 segments					
xv_time_segments_sort*	0,010000	0,020000	0,010000	0,010000	0,016000
34 segments					
xv_time_segments_merge*	0,002100	0,001700	0,001400	0,004900	0,005620
34 segments					
xv_time_segments_delta*	11,080000	11,770000	7,980000	12,340000	22,688000
34 segments					
xv_multizones_vis_time*	241,000000	214,000000	245,000000	423,000000	3773,600098
30 orbits					
xv_gen_scf*	2,720000	3,020000	2,870000	5,730000	38,063000
27 segments					
xv_multistations_vis_time*	418,000000	397,000000	421,000000	628,000000	12362,500000 0
10 orbits					
xv_station_vis_time*	58,000000	57,000000	58,000000	88,000000	2862,600098
10 orbits					
xv_station_vis_time_no_file*	9,000000	12,000000	10,000000	12,000000	29,700001
10 orbits					
xv_drs_vis_time*	336,000000	290,000000	188,000000	459,000000	536,000000
10 orbits					
xv_star_vis_time*	576,000000	560,000000	528,000000	835,000000	15609,500000 0
100 orbits					



xv_time_segments_mapping *	263,000000	249,000000	239,000000	401,000000	7307,799805
50 orbits					
xv_orbit_extra	22,600000	22,600000	21,000000	22,700001	49,680000
xv_gen_swath	265,399994	330,299988	274,200012	425,799988	4596,100098
xv_gen_swath_no_file	140,600006	190,199997	161,000000	219,800003	497,649994
xv_sc_vis_time *	196,000000	256,000000	219,000000	258,000000	610,900024
10 orbits					

Note that when the value “0.000000” is defined for a function in a certain platform, it means that its running time is lower than 1 nano-second and so it can be considered as “0”.**LIBRARY PRECAUTIONS**

The following precaution shall be taking into account when using EO\_VISIBILITY library:

- When a message like  
 <LIBRARY NAME> >>> ERROR in *xv\_function*: Internal computation error # *n*  
 or  
 <LIBRARY NAME> >>> WARNING in *xv\_function*: Internal computation warning # *n*  
 appears, run the program in **verbose** mode for a complete description of warnings and errors and call for maintenance if necessary.

## 9 KNOWN PROBLEMS

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

*Table 65: Known problems*

CFI library	Problem	Work around solution
xv_gps_vis_time	Functions not available yet	-