Earth Observation
Mission CFI Software

EO_ORBIT
SOFTWARE USER MANUAL

Code:	EO-MA-DMS-GS-0004
Issue:	4.1
Date:	07/05/10

	Name	Function	Signature
Prepared by:	Mariano Sánchez-Nogales	Project Engineer	
	Sara Cuenda Cuenda	Project Engineer	
	José Antonio González Abeytua	Project Manager	
	Juan Jose Borrego Bote	Project Engineer	
Checked by:	José Antonio González Abeytua	Project Manager	
Approved by:	José Antonio González Abeytua	Project Manager	

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

DOCUMENT INFORMATION

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

External Distribution		
Name	Organisation	Copies

Electronic handling	
Word Processor:	Adobe Framemaker 6.0 / OpenOffice 2.0
Archive Code:	P/SUM/DMS/01/026-011
Electronic file name:	eo-ma-dms-gs-004-21

DOCUMENT STATUS LOG

Issue	Change Description	Date	Approval
1.0	New document	08/11/01	
1.1	• New xo_orbit_to_time, xo_time_to_orbit and xo_free_osf_records functions. • xo_cart_extra removal	23/05/02	
1.2 Draft	• Cosmetic changes • Updated error handling	19/07/02	
2.0	Maintenance release.	29/11/02	
2.1	Maintenance release.	13/05/03	
2.2	• New options for xo_propag_init_file and xo_interpol_init_file functions. • Maintenance release.	30/09/03	
2.2.2	• Option to use a simplified algorithm to initialise using xo_propag_init_def • Absolute orbit and time since ANX calculated within xo_propag_extra and xo_interpol_extra functions. • Nodal period calculated by xo_orbit_info_from_<source> functions • Use of enumerations to size extra results arrays	26/04/04	
3.0	• New initialisation strategy for orbit calculations, propagation and interpolation. • New interfaces	21/07/04	
3.1	Maintenance release	13/10/04	
3.2	Maintenance release	15/11/04	
3.3	• Maintenance release • New features: – Changes for dealing with the new library explorer_data_handling – Identifier accessors. – Support for ENVISAT ASCII files removed	11/07/05	

3.4	• Maintenance release • New features: – Orbit file generation functions moved to this library	18/11/05	
3.5	• Maintenance release • New features: – time-orbit conversion executable – Support for SWARM and EARTHCARE	26/05/06	
3.6	• Maintenance release • New features: – xo_gen_oef – xo_check_osf and xo_check_oef	24/11/06	
3.7	• Maintenance release • New features: – Function expcfi_check_libs – Library version for MAC OS X on Intel (32 and 64-bits)	13/07/06	
3.7.2	• Maintenance release • New features: – TLE data for orbit operations and propagation – New executable: gen_oef	31/07/08	
4.0	• Maintenance release • New features: – Numerical propagator – New interfaces for propagation/interpolation	19/01/09	
4.1	• Maintenance release • New features: - Time correlation compatibility check between time_id and orbit file data	07/05/10	

TABLE OF CONTENTS

DOCUMENT INFORMATION 2

DOCUMENT STATUS LOG 3

TABLE OF CONTENTS 5

LIST OF TABLES 18

LIST OF FIGURES 23

1 SCOPE 24

2 ACRONYMS, NOMENCLATURE AND TERMINOLOGY 25

2.1 Acronyms 25

2.2 Nomenclature 25

2.3 Note on Terminology 26

3 APPLICABLE AND REFERENCE DOCUMENTS 27

3.1 Applicable Documents 27

3.2 Reference Documents 27

4 INTRODUCTION 28

4.1 Functions Overview 28

4.1.1 Orbit Initialisation 28

4.1.2 State Vector Computation (Propagation/Interpolation) 30

4.1.3 Ancillary Results Computation 31

4.1.4 Time/Orbit Transformation 31

4.1.5 Orbit Information Parameters 31

4.1.6 File Generation 31

4.1.7 Clean-up Memory 31

4.1.8 Check Orbit files 32

4.2 State Vector Computation Calling Sequence (Propagation/ Interpolation)..... 33

4.3 Time/Orbit Transformation and Orbit Information Parameters Calling Sequence..... 33

4.4 File Generation Calling Sequence..... 35

5 LIBRARY INSTALLATION..... 36

6 LIBRARY USAGE..... 37

6.1 Usage hints..... 39

6.2 General enumerations..... 40

6.3 Data Structures..... 43

7 CFI FUNCTIONS DESCRIPTION..... 46

7.1 xo_orbit_init_def..... 47

7.1.1 Overview..... 47

7.1.2 Calling interface..... 48

7.1.3 Input parameters..... 49

7.1.4 Output parameters..... 50

7.1.5 Warnings and errors..... 51

7.1.6 Runtime performances..... 51

7.2 xo_orbit_cart_init..... 52

7.2.1 Overview..... 52

7.2.2 Calling interface..... 52

7.2.3 Input parameters..... 53

7.2.4 Output parameters..... 53

7.2.5 Warnings and errors..... 54

7.2.6 Runtime performances..... 54

7.3 xo_orbit_cart_init_precise..... 56

7.3.1 Overview..... 56

7.3.2 Calling interface..... 56

7.3.3 Input parameters..... 57

7.3.4 Output parameters..... 58

7.3.5 Warnings and errors..... 59

7.3.6 Runtime performances 59

7.4 xo_orbit_init_file 60

7.4.1 Overview 60

7.4.2 Calling interface 61

7.4.3 Input parameters 62

7.4.4 Output parameters 63

7.4.5 Warnings and errors 64

7.4.6 Runtime performances 65

7.5 xo_orbit_init_file_precise 66

7.5.1 Overview 66

7.5.2 Calling interface 67

7.5.3 Input parameters 68

7.5.4 Output parameters 69

7.5.5 Warnings and errors 70

7.5.6 Runtime performances 70

7.6 xo_orbit_close 72

7.6.1 Overview 72

7.6.2 Calling interface 72

7.6.3 Input parameters 72

7.6.4 Output parameters 72

7.6.5 Warnings and errors 73

7.6.6 Runtime performances 73

7.7 xo_orbit_get_osv 74

7.7.1 Overview 74

7.7.2 Calling interface 74

7.7.3 Input parameters 74

7.7.4 Output parameters 74

7.7.5 Warnings and errors 75

7.7.6 Runtime performances 75

7.8 xo_orbit_set_osv 76

7.8.1 Overview 76

7.8.2 Calling interface 76

7.8.3 Input parameters..... 76
7.8.4 Output parameters..... 76
7.8.5 Warnings and errors..... 77
7.8.6 Runtime performances..... 77

7.9 xo_orbit_get_anx..... 78
7.9.1 Overview..... 78
7.9.2 Calling interface..... 78
7.9.3 Input parameters..... 78
7.9.4 Output parameters..... 78
7.9.5 Warnings and errors..... 79
7.9.6 Runtime performances..... 79

7.10 xo_orbit_set_anx..... 80
7.10.1 Overview..... 80
7.10.2 Calling interface..... 80
7.10.3 Input parameters..... 80
7.10.4 Output parameters..... 80
7.10.5 Warnings and errors..... 81
7.10.6 Runtime performances..... 81

7.11 xo_orbit_get_osf_rec..... 82
7.11.1 Overview..... 82
7.11.2 Calling interface..... 82
7.11.3 Input parameters..... 82
7.11.4 Output parameters..... 82
7.11.5 Warnings and errors..... 83
7.11.6 Runtime performances..... 83

7.12 xo_orbit_set_osf_rec..... 84
7.12.1 Overview..... 84
7.12.2 Calling interface..... 84
7.12.3 Input parameters..... 84
7.12.4 Output parameters..... 84
7.12.5 Warnings and errors..... 85
7.12.6 Runtime performances..... 85

7.13 xo_orbit_get_val_time 86
7.13.1 Overview 86
7.13.2 Calling interface 86
7.13.3 Input parameters 86
7.13.4 Output parameters 86
7.13.5 Warnings and errors 87
7.13.6 Runtime performances 87

7.14 xo_orbit_set_val_time 88
7.14.1 Overview 88
7.14.2 Calling interface 88
7.14.3 Input parameters 88
7.14.4 Output parameters 88
7.14.5 Warnings and errors 89
7.14.6 Runtime performances 89

7.15 xo_orbit_get_precise_propag_config 90
7.15.1 Overview 90
7.15.2 Calling interface 90
7.15.3 Input parameters 90
7.15.4 Output parameters 90
7.15.5 Warnings and errors 91
7.15.6 Runtime performances 91

7.16 xo_orbit_set_precise_propag_config 92
7.16.1 Overview 92
7.16.2 Calling interface 92
7.16.3 Input parameters 92
7.16.4 Output parameters 92
7.16.5 Warnings and errors 93
7.16.6 Runtime performances 93

7.17 xo_orbit_get_time_id 94
7.17.1 Overview 94
7.17.2 Calling interface 94
7.17.3 Input parameters 94

7.17.4 Output parameters..... 94
7.17.5 Warnings and errors..... 94
7.17.6 Runtime performances..... 95

7.18 xo_orbit_get_model_id..... 96
7.18.1 Overview..... 96
7.18.2 Calling interface..... 96
7.18.3 Input parameters..... 96
7.18.4 Output parameters..... 96
7.18.5 Warnings and errors..... 96
7.18.6 Runtime performances..... 97

7.19 xo_orbit_get_osv_compute_validity..... 98
7.19.1 Overview..... 98
7.19.2 Calling interface..... 98
7.19.3 Input parameters..... 98
7.19.4 Output parameters..... 98
7.19.5 Warnings and errors..... 99
7.19.6 Runtime performances..... 99

7.20 xo_orbit_get_propag_mode..... 100
7.20.1 Overview..... 100
7.20.2 Calling interface..... 100
7.20.3 Input parameters..... 100
7.20.4 Output parameters..... 100
7.20.5 Warnings and errors..... 101
7.20.6 Runtime performances..... 101

7.21 xo_orbit_get_interpol_mode..... 102
7.21.1 Overview..... 102
7.21.2 Calling interface..... 102
7.21.3 Input parameters..... 102
7.21.4 Output parameters..... 102
7.21.5 Warnings and errors..... 103
7.21.6 Runtime performances..... 103

7.22 xo_orbit_get_propag_config..... 104

7.22.1 Overview ... 104
7.22.2 Calling interface ... 104
7.22.3 Input parameters ... 104
7.22.4 Output parameters ... 104
7.22.5 Warnings and errors ... 105
7.22.6 Runtime performances ... 105

7.23 xo_orbit_get_interpol_config ... 106
7.23.1 Overview ... 106
7.23.2 Calling interface ... 106
7.23.3 Input parameters ... 106
7.23.4 Output parameters ... 106
7.23.5 Warnings and errors ... 107
7.23.6 Runtime performances ... 107

7.24 xo_orbit_id_clone ... 108
7.24.1 Overview ... 108
7.24.2 Calling interface ... 108
7.24.3 Input parameters ... 108
7.24.4 Output parameters ... 108
7.24.5 Warnings and errors ... 108
7.24.6 Runtime performances ... 109

7.25 xo_run_init ... 110
7.25.1 Overview ... 110
7.25.2 Calling interface ... 110
7.25.3 Input parameters ... 110
7.25.4 Output parameters ... 110
7.25.5 Warnings and errors ... 111
7.25.6 Runtime performances ... 111

7.26 xo_run_get_ids ... 112
7.26.1 Overview ... 112
7.26.2 Calling interface ... 112
7.26.3 Input parameters ... 112
7.26.4 Output parameters ... 112

7.26.5 Warnings and errors..... 113
7.26.6 Runtime performances..... 113

7.27 xo_run_close..... 114
7.27.1 Overview..... 114
7.27.2 Calling interface..... 114
7.27.3 Input parameters..... 114
7.27.4 Output parameters..... 114
7.27.5 Warnings and errors..... 114
7.27.6 Runtime performances..... 115

7.28 xo_osv_compute..... 116
7.28.1 Overview..... 116
7.28.2 Computation methods (Propagation/interpolation)..... 116
7.28.2.1 Propagation methods..... 117
7.28.2.2 Interpolation methods..... 120
7.28.3 Calling interface..... 122
7.28.4 Input parameters..... 122
7.28.5 Output parameters..... 123
7.28.6 Warnings and errors..... 124
7.28.7 Runtime performances..... 124

7.29 xo_osv_compute_extra..... 125
7.29.1 Overview..... 125
7.29.2 Calling interface..... 125
7.29.3 Input parameters..... 126
7.29.4 Output parameters..... 127
7.29.5 Results vectors..... 128
7.29.6 Warnings and errors..... 133
7.29.7 Runtime performances..... 133

7.30 xo_orbit_to_time..... 134
7.30.1 Overview..... 134
7.30.2 Calling sequence of xo_orbit_to_time:..... 134
7.30.3 Input parameters..... 135
7.30.4 Output parameters..... 135

7.30.5 Warnings and errors..... 135
7.30.6 Runtime performances..... 136
7.30.7 Executable Program..... 136

7.31 xo_time_to_orbit..... 138
7.31.1 Overview..... 138
7.31.2 Calling sequence of xo_time_to_orbit..... 138
7.31.3 Input parameters..... 139
7.31.4 Output parameters..... 139
7.31.5 Warnings and errors..... 139
7.31.6 Runtime performances..... 140
7.31.7 Executable Program..... 141

7.32 xo_orbit_info..... 143
7.32.1 Overview..... 143
7.32.2 Calling sequence of xo_orbit_info..... 143
7.32.3 Input parameters..... 144
7.32.4 Output parameters..... 144
7.32.5 Warnings and errors..... 145
7.32.6 Runtime performances..... 145

7.33 xo_orbit_rel_from_abs..... 146
7.33.1 Overview..... 146
7.33.2 Calling sequence of xo_orbit_rel_from_abs..... 146
7.33.3 Input parameters..... 147
7.33.4 Output parameters..... 147
7.33.5 Warnings and errors..... 148
7.33.6 Runtime performances..... 148

7.34 xo_orbit_abs_from_rel..... 149
7.34.1 Overview..... 149
7.34.2 Calling sequence of xo_orbit_abs_from_rel..... 149
7.34.3 Input parameters..... 150
7.34.4 Output parameters..... 150
7.34.5 Warnings and errors..... 151
7.34.6 Runtime performances..... 151

7.35 xo_orbit_abs_from_phase 152
7.35.1 Overview 152
7.35.2 Calling sequence of xo_orbit_abs_from_phase 152
7.35.3 Input parameters 153
7.35.4 Output parameters 153
7.35.5 Warnings and errors 154
7.35.6 Runtime performances 154

7.36 xo_osv_to_tle 155
7.36.1 Overview 155
7.36.2 Calling sequence of xo_osv_to_tle 155
7.36.3 Input parameters 156
7.36.4 Output parameters 156
7.36.5 Warnings and errors 157
7.36.6 Runtime performances 157

7.37 xo_gen_osf_create 158
7.37.1 Overview 158
7.37.2 Calling interface 158
7.37.3 Input parameters 160
7.37.4 Output parameters 161
7.37.5 Warnings and errors 162
7.37.6 Runtime performances 163
7.37.7 Executable Program 164

7.38 xo_gen_osf_append_orbit_change 166
7.38.1 Overview 166
7.38.2 Calling interface 166
7.38.3 Input parameters 168
7.38.4 Output parameters 169
7.38.5 Warnings and errors 170
7.38.6 Runtime performances 171
7.38.7 Executable Program 172

7.39 xo_gen_osf_change_repeat_cycle 174
7.39.1 Overview 174

7.39.2 Calling interface..... 174
7.39.3 Input parameters..... 176
7.39.4 Output parameters..... 177
7.39.5 Warnings and errors..... 178
7.39.6 Runtime performances..... 179
7.39.7 Executable Program..... 180

7.40 xo_gen_osf_add_drift_cycle..... 182
7.40.1 Overview..... 182
7.40.2 Calling interface..... 182
7.40.3 Input parameters..... 184
7.40.4 Output parameters..... 185
7.40.5 Warnings and errors..... 185
7.40.6 Runtime performances..... 187
7.40.7 Executable Program..... 188

7.41 xo_gen_rof..... 190
7.41.1 Overview..... 190
7.41.2 Calling interface..... 190
7.41.3 Input parameters..... 192
7.41.4 Output parameters..... 195
7.41.5 Warnings and errors..... 195
7.41.6 Runtime performances..... 196
7.41.7 Executable Program..... 197

7.42 xo_gen_rof_prototype..... 199
7.42.1 Overview..... 199
7.42.2 Calling interface..... 199
7.42.3 Input parameters..... 201
7.42.4 Output parameters..... 203
7.42.5 Warnings and errors..... 203
7.42.6 Runtime performances..... 204

7.43 xo_gen_pof..... 205
7.43.1 Overview..... 205
7.43.2 Calling interface..... 205

7.43.3 Input parameters..... 207
7.43.4 Output parameters..... 208
7.43.5 Warnings and errors..... 209
7.43.6 Runtime performances..... 210
7.43.7 Executable Program..... 211

7.44 xo_gen_oef..... 213
7.44.1 Ov erview..... 213
7.44.2 Calling interface..... 213
7.44.3 Input parameters..... 213
7.44.4 Output parameters..... 214
7.44.5 Warnings and errors..... 214
7.44.6 Runtime performances..... 215
7.44.7 Executable Program..... 216

7.45 xo_gen_dnf..... 217
7.45.1 Overview..... 217
7.45.2 Calling interface..... 217
7.45.3 Input parameters..... 219
7.45.4 Output parameters..... 221
7.45.5 Warnings and errors..... 222
7.45.6 Runtime performances..... 223

7.46 xo_gen_tle..... 226
7.46.1 Overview..... 226
7.46.2 Calling interface..... 226
7.46.3 Input parameters..... 227
7.46.4 Output parameters..... 228
7.46.5 Warnings and errors..... 228
7.46.6 Runtime performances..... 229

7.47 xo_check_osf..... 230
7.47.1 Overview..... 230
7.47.2 Calling interface..... 230
7.47.3 Input parameters..... 231
7.47.4 Output parameters..... 231

7.47.5 Warnings and errors..... 232
7.47.6 Runtime performances..... 233

7.48 xo_check_oef..... 234
7.48.1 Overview..... 234
7.48.2 Calling interface..... 234
7.48.3 Input parameters..... 235
7.48.4 Output parameters..... 236
7.48.5 Warnings and errors..... 237
7.48.6 Runtime performances..... 237

8 LIBRARY PRECAUTIONS..... 238

9 KNOWN PROBLEMS..... 239

LIST OF TABLES

Table 1: CFI functions included within EO_ORBIT library..... 38
Table 2: Some enumerations within EO_ORBIT library..... 40
Table 3: EO_ORBIT structures..... 43
Table 4: Input parameters of xo_orbit_init_def function..... 49
Table 5: Output parameters of xo_orbit_init_def function..... 50
Table 6: Error messages of xo_orbit_init_def function..... 51
Table 7: Runtime performances of xo_orbit_init_def function..... 51
Table 8: Input parameters of xo_orbit_cart_init function..... 53
Table 9: Output parameters of xo_orbit_cart_init function..... 53
Table 10: Error messages of xo_orbit_cart_init function..... 54
Table 11: Runtime performances of xo_orbit_cart_init function..... 55
Table 12: Input parameters of xo_orbit_cart_init_precise function..... 57
Table 13: Output parameters of xo_orbit_cart_init_precise function..... 58
Table 14: Error messages of xo_orbit_cart_init_precise function..... 59
Table 15: Runtime performances of xo_orbit_cart_init_precise function..... 59
Table 16: User requested time range in xo_orbit_init_file..... 60
Table 17: Validity periods for xo_orbit_init_file..... 61
Table 18: Input parameters of xo_orbit_init_file function..... 62
Table 19: Output parameters of xo_orbit_init_file function..... 63
Table 20: Error messages of xo_orbit_init_file function..... 64
Table 21: Runtime performances of xo_orbit_init_file function..... 65
Table 22: User requested time range in xo_orbit_init_file_precise..... 66
Table 23: Validity periods for xo_orbit_init_file_precise..... 66
Table 24: Input parameters of xo_orbit_init_file_precise function..... 68
Table 25: Output parameters of xo_orbit_init_file_precise function..... 69
Table 26: Error messages of xo_orbit_init_file_precise function..... 70
Table 27: Runtime performances of xo_orbit_init_file_precise function..... 71
Table 28: Input parameters of xo_orbit_close function..... 72
Table 29: Output parameters of xo_orbit_close function..... 72
Table 30: Error messages of xo_orbit_close function..... 73

Table 31: Input parameters of xo_orbit_get_osv function..... 74
Table 32: Output parameters of xo_orbit_get_osv function..... 74
Table 33: Input parameters of xo_orbit_set_osv function..... 76
Table 34: Output parameters of xo_orbit_set_osv function..... 77
Table 35: Input parameters of xo_orbit_get_anx function..... 78
Table 36: Output parameters of xo_orbit_get_anx function..... 79
Table 37: Input parameters of xo_orbit_set_anx function..... 80
Table 38: Output parameters of xo_orbit_set_anx function..... 81
Table 39: Input parameters of xo_orbit_get_osf_rec function..... 82
Table 40: Output parameters of xo_orbit_get_osf_rec function..... 82
Table 41: Runtime performances of xo_orbit_get_osf_rec function..... 83
Table 42: Input parameters of xo_orbit_set_osf_rec function..... 84
Table 43: Output parameters of xo_orbit_set_osf_rec function..... 85
Table 44: Runtime performances of xo_orbit_set_osf_rec function..... 85
Table 45: Input parameters of xo_orbit_get_val_time function..... 86
Table 46: Output parameters of xo_orbit_get_val_time function..... 86
Table 47: Input parameters of xo_orbit_set_val_time function..... 88
Table 48: Output parameters of xo_orbit_set_val_time function..... 88
Table 49: Runtime performances of xo_orbit_set_val_time function..... 89
Table 50: Input parameters of xo_orbit_get_precise_propag_config function..... 90
Table 51: Output parameters of xo_orbit_get_precise_propag_config function..... 90
Table 52: Runtime performances of xo_orbit_get_precise_propag_config function..... 91
Table 53: Input parameters of xo_orbit_set_precse_propag_config function..... 92
Table 54: Output parameters of xo_orbit_set_precse_propag_config function..... 92
Table 55: Runtime performances of xo_orbit_set_precse_propag_config function..... 93
Table 56: Input parameters of xo_orbit_get_time_id function..... 94
Table 57: Output parameters of xo_orbit_get_time_id function..... 94
Table 58: Input parameters of xo_orbit_get_model_id function..... 96
Table 59: Output parameters of xo_orbit_get_model_id function..... 96
Table 60: Input parameters of xo_orbit_get_osv_compute_validity function..... 98
Table 61: Output parameters of xo_orbit_get_osv_compute_validity function..... 98
Table 62: Input parameters of xo_orbit_get_propag_mode function..... 100
Table 63: Output parameters of xo_orbit_get_propag_mode function..... 100
Table 64: Input parameters of xo_orbit_get_interpol_mode function..... 102

Table 65: Output parameters of xo_orbit_get_interpol_mode function..... 102
Table 66: Input parameters of xo_orbit_get_propag_config function..... 104
Table 67: Output parameters of xo_orbit_get_propag_config function..... 104
Table 68: Input parameters of xo_orbit_get_interpol_config function..... 106
Table 69: Output parameters of xo_orbit_get_interpol_config function..... 106
Table 70: Input parameters of xo_orbit_id_clone function..... 108
Table 71: Output parameters of xo_orbit_id_clone function..... 108
Table 72: Input parameters of xo_run_init function..... 110
Table 73: Output parameters of xo_run_init function..... 110
Table 74: Error messages of xo_run_init function..... 111
Table 75: Input parameters of xo_run_get_ids function..... 112
Table 76: Output parameters of xo_run_get_ids function..... 112
Table 77: Input parameters of xo_run_close function..... 114
Table 78: Output parameters of xo_run_close function..... 114
Table 79: OSV computation methods..... 116
Table 80: Validity Time Intervals for Propagation..... 119
Table 81: Input parameters of xo_osv_compute function..... 122
Table 82: Output parameters of xo_osv_compute function..... 123
Table 83: Error messages of xo_osv_compute function..... 124
Table 84: Runtime performances of xo_osv_compute function..... 124
Table 85: Input parameters of xo_osv_compute_extra..... 126
Table 86: Enumeration values of extra_choice input flag..... 126
Table 87: Output parameters of xo_osv_compute_extra..... 127
Table 88: Ancillary results vector. Model-dependent parameters..... 128
Table 89: Ancillary results vector. Model-independent parameters..... 129
Table 90: Error messages of xo_osv_compute_extra function..... 133
Table 91: Runtime performances of xo_osv_compute_extra function..... 133
Table 92: Input parameters for xo_orbit_to_time..... 135
Table 93: Output parameters for xo_orbit_to_time..... 135
Table 94: Error messages of xo_orbit_to_time function..... 136
Table 95: Runtime performances of xo_orbit_to_time function..... 136
Table 96: Input parameters for xo_time_to_orbit function..... 139
Table 97: Output parameters for xo_time_to_orbit..... 139
Table 98: Error messages of xo_time_to_orbit function..... 140

Table 99: Runtime performances of xo_time_to_orbit function..... 140
Table 100: Input parameters for xo_orbit_info..... 144
Table 101: Output parameters for xo_orbit_info..... 144
Table 102: Error messages of xo_orbit_info function..... 145
Table 103: Runtime performances of xo_orbit_info function..... 145
Table 104: Input parameters for xo_orbit_rel_from_abs..... 147
Table 105: Output parameters for xo_orbit_rel_from_abs..... 147
Table 106: Error messages of xo_orbit_rel_from_abs function..... 148
Table 107: Runtime performances of xo_orbit_rel_from_abs function..... 148
Table 108: Input parameters for xo_orbit_abs_from_rel..... 150
Table 109: Output parameters for xo_orbit_abs_from_rel..... 150
Table 110: Error messages of xo_orbit_abs_from_rel function..... 151
Table 111: Runtime performances of xo_orbit_abs_from_rel function..... 151
Table 112: Input parameters for xo_orbit_abs_from_phase..... 153
Table 113: Output parameters for xo_orbit_abs_from_phase..... 153
Table 114: Error messages of xo_orbit_abs_from_phase function..... 154
Table 115: Runtime performances of xo_orbit_abs_from_phase function..... 154
Table 116: Input parameters for xo_osv_to_tle..... 156
Table 117: Output parameters for xo_osv_to_tle..... 156
Table 118: Error messages of xo_osv_to_tle function..... 157
Table 119: Runtime performances of xo_osv_to_tle function..... 157
Table 120: Input parameters of xo_gen_osf_create function..... 160
Table 121: Output parameters of xo_gen_osf_create function..... 161
Table 122: Error messages of xo_gen_osf_create function..... 162
Table 123: Runtime performances of xo_gen_osf_create function..... 163
Table 124: Input parameters of xo_gen_osf_append_orbit_change function..... 168
Table 125: Output parameters of xo_gen_osf_append_orbit_change function..... 169
Table 126: Error messages of xo_gen_osf_append_orbit_change function..... 170
Table 127: Runtime performances of xo_gen_osf_append_orbit_change function..... 171
Table 128: Input parameters of xo_gen_osf_change_repeat_cycle function..... 176
Table 129: Output parameters of xo_gen_osf_change_repeat_cycle function..... 177
Table 130: Error messages of xo_gen_osf_change_repeat_cycle function..... 178
Table 131: Runtime performances of xo_gen_osf_change_repeat_cycle function..... 179
Table 132: Input parameters of xo_gen_osf_add_drift_cycle function..... 184

Table 133: Output parameters of xo_gen_osf_add_drift_cycle function..... 185
Table 134: Error messages of xo_gen_osf_add_drift_cycle function..... 186
Table 135: Runtime performances of xo_gen_osf_add_drift_cycle function..... 187
Table 136: Input parameters of xo_gen_rof function..... 192
Table 137: Output parameters of xo_gen_rof function..... 195
Table 138: Error messages of xo_gen_rof function..... 195
Table 139: Runtime performances of xo_gen_rof function..... 196
Table 140: Input parameters of xo_gen_rof_prototype function..... 201
Table 141: Output parameters of xo_gen_rof_prototype function..... 203
Table 142: Error messages of xo_gen_rof_prototype function..... 203
Table 143: Runtime performances of xo_gen_rof_prototype function..... 204
Table 144: Input parameters of xo_gen_pof function..... 207
Table 145: Output parameters of xo_gen_pof function..... 208
Table 146: Error messages of xo_gen_pof function..... 209
Table 147: Runtime performances of xo_gen_pof function..... 210
Table 148: Input parameters of xo_gen_oef function..... 213
Table 149: Output parameters of xo_gen_oef function..... 214
Table 150: Error messages of xo_gen_oef function..... 214
Table 151: Runtime performances of xo_gen_oef function..... 215
Table 152: Input parameters of xo_gen_dnf function..... 219
Table 153: Output parameters of xo_gen_dnf function..... 221
Table 154: Error messages of xo_gen_dnf function..... 222
Table 155: Runtime performances of xo_gen_dnf function..... 223
Table 156: Input parameters of xo_gen_tle function..... 227
Table 157: Output parameters of xo_gen_tle function..... 228
Table 158: Error messages of xo_gen_tle function..... 228
Table 159: Runtime performances of xo_gen_tle function..... 229
Table 160: Input parameters of xo_check_osf function..... 231
Table 161: Output parameters of xo_check_osf function..... 231
Table 162: Error messages of xo_ckeck_osf function..... 232
Table 163: Runtime performances of xo_check_osf function..... 233
Table 164: Input parameters of xo_check_oef function..... 235
Table 165: Output parameters of xo_check_oef function..... 236
Table 166: Error messages of xo_ckeck_oef function..... 237

Table 167: Runtime performances of xo_check_oef function..... 237
Table 168: Known problems list..... 239

LIST OF FIGURES

Figure 1: Orbit Calling Sequence..... 34
Figure 2: File Generation Calling Sequence..... 35
Figure 3: Weight Function for Double Propagation Model..... 118
Figure 4: Performances of the interpolation algorithm..... 121

1 SCOPE

The EO_ORBIT Software User Manual provides a detailed description of usage of the CFI functions included within the EO_ORBIT 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
EOCFI	Earth Observation CFI
ESA	European Space Agency
ESTEC	European Space Technology and Research Centre
FOS	Flight Operations Segment
GS	Ground Station
OBT	On-board Binary Time
OEF	Orbit Event File
OSF	Orbit Scenario File
OSV	Orbit State Vector
POF	Predicted Orbit File
ROF	Restituted Orbit File
SSP	Sub-Satellite Point
SRAR	Satellite Relative Actual Reference
SUM	Software User Manual
TLE	Two Line Elements
TOD	True of Date reference frame
UTC	Universal Time Coordinated
UT1	Universal Time UT1
WGS[84]	World Geodetic System 1984

2.2 Nomenclature

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

2.3 Note on Terminology

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

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

3 APPLICABLE AND REFERENCE DOCUMENTS

3.1 Applicable Documents

[GEN_SUM] Earth Observation Mission CFI Software. General Software User Manual. EO-MA- DMS-GS-0002. Issue 4.1 07/05/10.

3.2 Reference Documents

[MCD] Earth Observation Mission CFI Software. Conventions Document. EO-MA-DMS-GS-0001. Issue 1.0 27/10/09.

[MSC] Earth Observation Mission CFI Software. Mission Specific Customizations. EO-MA- DMS-GS-0018. Issue 1.0 27/10/09

[F_H_SUM] Earth Observation Mission CFI Software. EO_FILE_HANDLING Software User Manual. EO-MA-DMS-GS-0008. Issue 4.1 07/05/10.

[D_H_SUM] Earth Observation Mission CFI Software. EO_DATA_HANDLING Software User Manual. EO-MA-DMS-GS-007. Issue 4.1 07/05/10.

[LIB_SUM] Earth Observation Mission CFI Software. EO_LIB Software User Manual. EO-MA-DMS-GS-003. Issue 4.1 07/05/10.

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

4 INTRODUCTION

4.1 Functions Overview

This software library contains:

- CFI functions allowing accurate computation of orbit state vectors, either at ascending node or (by propagation) at any point in the orbit of any Earth Observation satellite.
- The orbit propagation may be performed based on different propagation models. The initial set of models supported are:
  - Mean Keplerian model
  - TLE model
  - Numerical model
  - Spot model
- It includes an interpolator and orbit propagators.
- CFI functions required to compute the orbit scenario file, used for Earth Observation mission planning purposes, and several orbit files useful for testing purposes (Predicted Orbit File, Restituted Orbit File, DORIS Navigator Files).
- It contains:
  - a library of functions which can be called from a main executable program
  - a set of executable programs (1 for each function) with the exact same functionality as the functions

The following sections summarize the set of functions in this library:

4.1.1 Orbit Initialisation

Before doing any orbit calculation, the orbit should be initialized using one of the following functions:

- ***xo_orbit_init_def***: this software generates a cartesian state vector around the true ascending node crossings as a function of the date (processing time), the longitude of the ascending node, the satellite Repeat Cycle Length, the mean local solar time and either the drift in mean local solar time or the inclination. For the Spot model, the routine generates the Spot elements.
- ***xo_orbit_cart_init, xo_orbit_cart_init_precise***: This software initializes the orbit using as input a cartesian orbit state vector. The "precise" function allows the introduction of data to propagate an state vector with a numeric propagator (see 4.1.2). Numerical propagator uses external files for the configuration of gravity model and F10.7 coefficient and Geomagnetic Activity index values. You can find some files that can be used in *files/models* directory of the Earth Observation CFI package, and following you can find some references for them:
  - Gravity model egm96:
  - http://cddisa.gsfc.nasa.gov/926/egm96/egm96.html
  - F10.7 coefficient and Geomagnetic Activity index: ECSS-E-10-04A Space Environment

– (http://www.spacelab.dti.supsi.ch/Tecnica/ECSS-E-10-04ASpaceEnvironment1.pdf).

- ***xo_orbit_init_file, xo_orbit_init_file_precise***: This software initializes the orbit using a set of files containing the orbital information (state vectors, orbital geometry or TLE data). The "precise" function allows the introduction of data to propagate an state vector with a numeric propagator (see 4.1.2). The following input file types are accepted:
  - Flight Dynamics predicted ascending node state vectors.
  - DORIS Navigator Data
  - FOS Restituted Orbit Files
  - DORIS Preliminary Orbit
  - DORIS Precise Orbit
  - Ascending node state vectors from the Orbit Scenario File
  - State vectors from Spot orbit files.
  - TLE files (not for precise propagator)

In all cases a variable of the type xo_orbit_id (*Orbit ID.*) is returned. This variable is a CFI Identifier of the type described in [GEN_SUM]. This variable keeps internally the orbit information that will be used for further calculations. That orbit information can be retrieved by calling the following CFI functions:

CFI Function[1]	Orbit ID data	Condition to get the data
xo_orbit_init_status	Orbit ID initialisation status	Always
xo_orbit_get_sat_id	Satellite ID	The Orbit ID is initialised.
xo_orbit_get_mode	Mode used for the Orbit ID initialisation	The Orbit ID is initialised.
xo_orbit_get_osv	OSV stored in the Orbit ID	The Orbit ID has bee initialised with state vectors.
xo_orbit_get_anx	ANX data stored in the Orbit ID	The Orbit ID has bee initialised with state vectors that are not located at the ANX (Restituted orbit files, DORIS Navigator files...)
xo_orbit_get_osf_rec	Orbital Geometry data stored in the Orbit ID	The Orbit ID has bee initialised with orbit geometry data.
xo_orbit_get_val_time	Validity time interval where the Orbit ID can be used except for **xo_osv_compute** and **xo_osv_compute_extra**	The Orbit ID is initialised.
xo_orbit_get_precise_propag_config	Configuration for the precise propagator	The Orbit ID has been initialised with **xo_orbit_init_file_precise** or **xo_orbit_cart_init_precise**
xo_orbit_get_time_id	Time ID used for the Orbit ID initialisation	The Orbit ID is initialised.

1 These functions are defined in the current SUM (section 7) or in [GEN_SUM].

xo_orbit_get_model_id	Model ID used for the Orbit ID initialisation	The Orbit ID is initialised.
xo_orbit_get_osv_compute_validity	Validity time interval where the Orbit ID can be used to call **xo_osv_compute** and **xo_osv_compute_extra**	The Orbit ID is initialised.
xo_orbit_get_propag_mode	Propagation model used when calling **xo_osv_compute**	Orbit Id is configured for propagation (see section 4.1.2)
xo_orbit_get_interpol_mode	Interpolation model used when calling **xo_osv_compute**	Orbit Id is configured for interpolation (see section 4.1.2)
xo_orbit_get_propag_config	Configuration data for the propagator	Orbit Id is configured for propagation (see section 4.1.2)
xo_orbit_get_interpol_config	Configuration data for the interpolator	Orbit Id is configured for interpolation (see section 4.1.2)

Finally, note that it is possible to create a copy of *Orbit ID* with **xo_orbit_id_clone** .

4.1.2 State Vector Computation (Propagation/Interpolation)

The software provides a set of functions to compute orbit state vectors at a given time:

- ***xo_osv_compute:*** This software computes the state vector at the requested time. The method used to compute that vector is transparent for the user and depends on the data type used for the orbit initialisation. Propagation is performed when the orbit_id is initialised with:
  - One Orbit State Vector (**xo_orbit_cart_init**)
  - Orbit Geometry (**xo_orbit_init_def**)
  - Orbit Scenario File
  - Predicted orbit file (plus an optional DORIS Navigator file)
  - Orbit Event Files
  - TLE files

Interpolation is used in these other cases:

- DORIS Navigator Data
- FOS Restituted Orbit Files
- DORIS Preliminary Orbit
- DORIS Precise Orbit

4.1.3 Ancillary Results Computation

- ***xo_osv_compute_extra***: This software returns ancillary results, i.e. mean and osculating Keplerian orbit state vectors, satellite osculating true latitude, latitude rate and latitude rate-rate, Sun zenith angle and many more.

4.1.4 Time/Orbit Transformation

- ***xo_time_to_orbit***: This software calculates the absolute orbit, number of seconds and number of microseconds since ascending node that corresponds to a given time in processing format.
- ***xo_orbit_to_time***: This software calculates the time, in processing format, that corresponds to a given absolute orbit, number of seconds and number of microseconds since ascending node.

4.1.5 Orbit Information Parameters

- ***xo_orbit_rel_from_abs***: This software calculates the relative orbit, the phase number giving as input an absolute orbit number.
- ***xo_orbit_abs_from_rel***: This software calculates the absolute orbit number giving as input a relative orbit number and its cycle number.
- ***xo_orbit_abs_from_phase***: This software calculates the absolute orbit number, the relative orbit, the phase number giving as input a phase number.
- ***xo_orbit_info***: This software calculates orbit related parameters providing as input the absolute orbit number.

4.1.6 File Generation

- ***xo_gen_osf_create***: generates the orbit scenario file with user provided inputs
- ***xo_gen_osf_append_orbit_change***: adds an orbit change to a previously generated OSF
- ***xo_gen_osf_change_repeat_cycle***: adds an orbit change for a given target orbit to an existing OSF.
- ***xo_gen_osf_add_drift_cycle***: adds an orbit change for a requested orbit with a particular ascending node longitude and an orbit for the manoeuvre.
- ***xo_gen_pof***: generates a Predicted Orbit File from several different reference input files.
- ***xo_gen_rof*** and ***xo_gen_rof_prototype***: generates a Restituted Orbit File from several different reference input files.
- ***xo_gen_oef*** generates an orbit event file from an orbit scenario file and a predicted orbit file.
- ***xo_gen_dnf***: generates a DORIS Navigator File from several different reference input files.
- ***xo_gen_tle***: generates a TLE file from a Predicted Orbit file.

4.1.7 Clean-up Memory

- ***xo_orbit_close***: This software frees the memory allocated by the orbit initialization routines. It closes the xo_orbit_id, so that it cannot be used for further computations.

4.1.8 Check Orbit files

- ***xo_check_osf***: checks the continuity between the last orbit of an orbital change and the next orbit in an orbit scenario file.
- ***xo_check_oef***: checks the consistency between the list of the orbital changes and the list of orbit state vectors in an orbit event file.

4.2 State Vector Computation Calling Sequence (Propagation/ Interpolation)

A complete propagation sequence consists of:

- A call to any of the initialization routines for orbit, ***xo_orbit_init_def***, ***xo_orbit_init_file[_precise]*** or ***xo_orbit_cart_init[_precise]***, to generate the internal data necessary for whatever calculation involving orbits.
- An optional call to ***xo_osv_compute_extra*** to calculate any desired ancillary result related to the initializing state vector.
- A call to the ***xo_osv_compute*** function to compute the orbit state vector at a requested time (Optionally, the user can check if the requested time is within the the validity interval by calling the function ***xo_orbit_get_osv_compute_validity***).
- To obtain some ancillary results associated to the computed OSV, the user might call the ***xo_osv_compute_extra*** function.
- At the end of a sequence is mandatory to call ***xo_orbit_close*** to free the memory allocated.

The possible propagation sequences of calls allowing to produce an orbit state vector are shown in Figure 1.

4.3 Time/Orbit Transformation and Orbit Information Parameters Calling Sequence

A complete time/orbit transformation and orbit information parameters sequence consists of:

- A call to any of the initialization routines for orbit, ***xo_orbit_init_def***, ***xo_orbit_init_file[_precise]*** or ***xo_orbit_cart_init[_precise]***, to generate the internal data necessary for whatever calculation involving orbits. Note that time to orbit transformations cannot be computed if the orbit was initialised with ***xo_orbit_cart_init***.
- A call to a ***time/orbit transformation*** or an ***orbit information parameters*** routine.
- When no more ***time/orbit transformations*** and ***orbit information parameters*** routines are going to be used, call to ***xo_orbit_close*** to free the memory allocated.
- The possible time/orbit transformation and orbit information parameters sequences of calls allowing to produce an orbit state vector are shown in Figure 1.
- A detailed description of each function is provided in section 7. Please refer also to:
- [MCD]for a detailed description of the time references and formats, reference frames, parameters and models used in this document.
- [GEN_SUM] for a complete overview of the CFI, and in particular the detailed description of the *Id* concept and the error handling functions.

Orbit Routines Data Flow

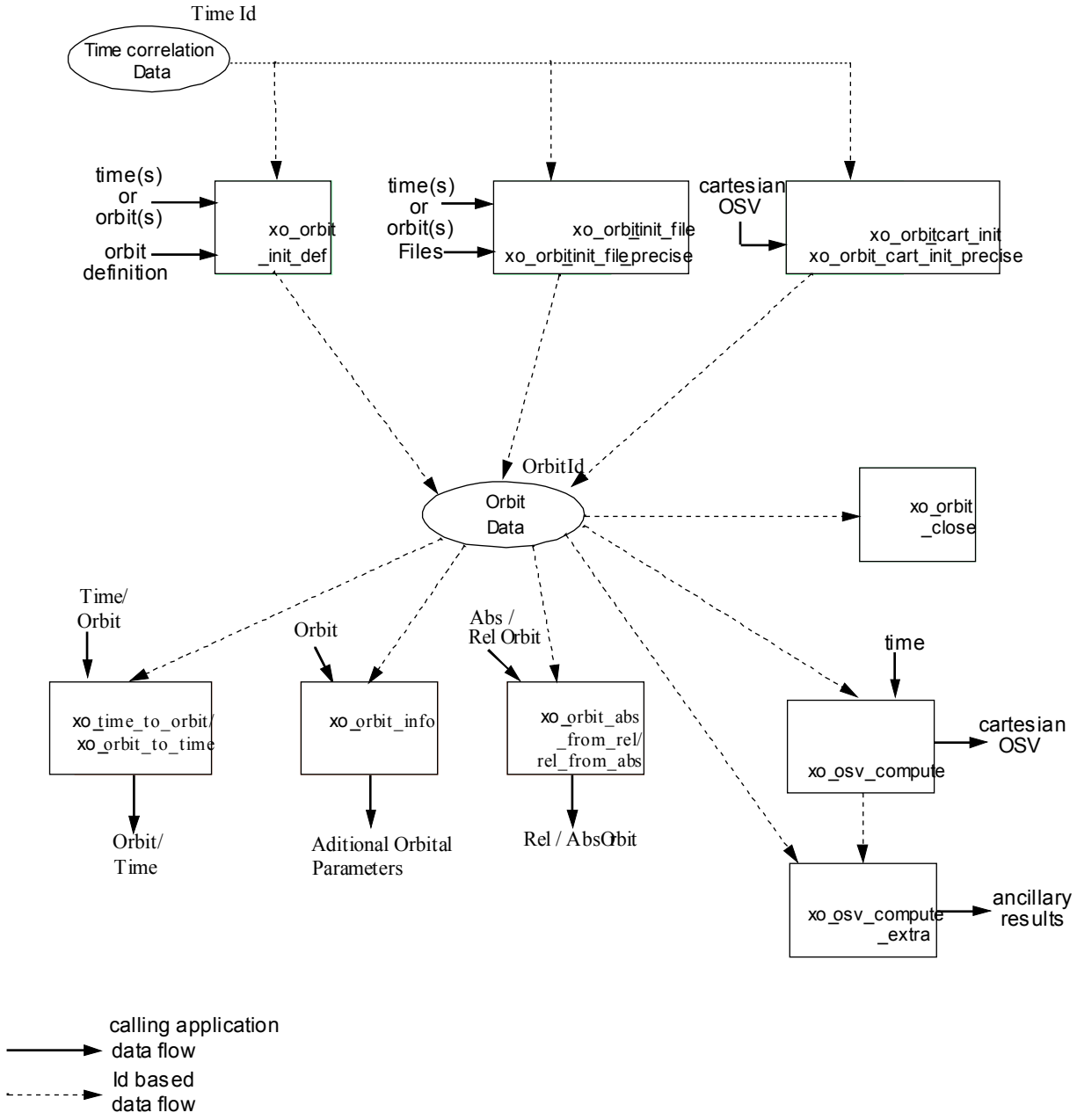


Figure 1: Orbit Calling Sequence

4.4 File Generation Calling Sequence

The calling sequence for the file generators consists of:

- One call to a time initialization routine
- One call to the generation routine providing the input parameters. For **xo_gen_pof**, **xo_gen_rof**, **xo_gen_oef** and **xo_gen_dnf** a reference orbit file has to be provided as well.

The following figure shows an schema of the calling sequence:

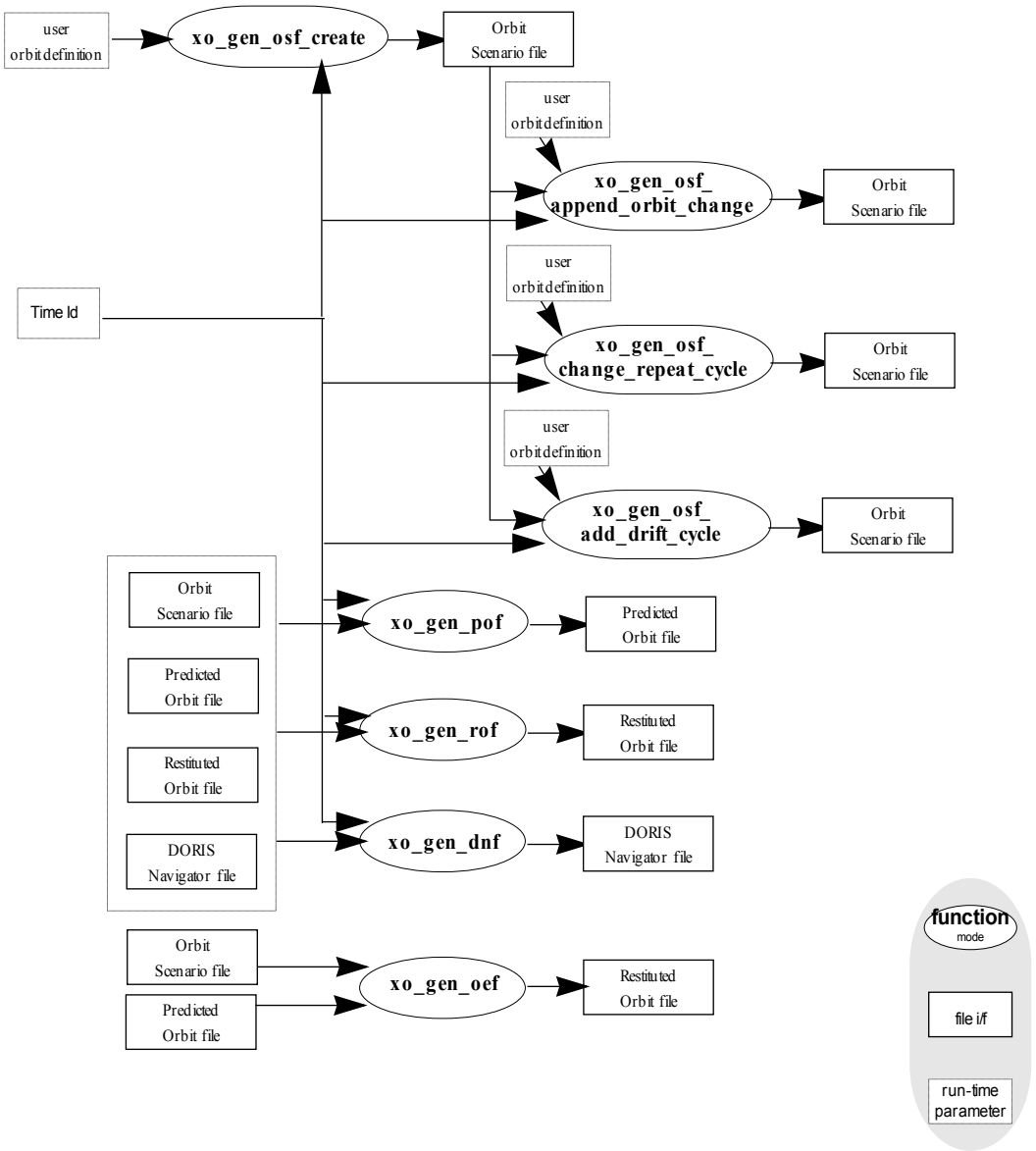


Figure 2: File Generation Calling Sequence

5 LIBRARY INSTALLATION

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

6 LIBRARY USAGE

Note that to use the EO_ORBIT 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])

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

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

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

- explorer_orbit.h (for a C application)

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

- SOLARIS/LINUX:

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

- WINDOWS:

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

- MacOS:

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

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

To avoid problems in linking a user application with the `EO_ORBIT` 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 XO_ or xo_.

This is summarized in Table 1.

Table 1: CFI functions included within EO_ORBIT library

Function Name	Enumeration value	long
Main CFI Functions		
xo_orbit_init_def	XO_ORBIT_INIT_DEF_ID	0
xo_orbit_cart_init	XO_ORBIT_CART_INIT_ID	1
xo_orbit_cart_init_precise	XO_ORBIT_CART_INIT_PRECISE_ID	2
xo_orbit_init_file	XO_ORBIT_INIT_FILE_ID	3
xo_orbit_init_file_precise	XO_ORBIT_INIT_FILE_PRECISE_ID	4
xo_orbit_close	XO_ORBIT_CLOSE_ID	5
xo_osv_compute	XO_OSV_COMPUTE_ID	6
xo_osv_compute_extra	XO_OSV_COMPUTE_EXTRA_ID	7
xo_orbit_to_time	XO_ORBIT_TO_TIME_ID	8
xo_time_to_orbit	XO_TIME_TO_ORBIT_ID	9
xo_orbit_abs_from_rel	XO_ORBIT_ABS_FROM_REL_ID	10
xo_orbit_rel_from_abs	XO_ORBIT_REL_FROM_ABS_ID	11
xo_orbit_abs_from_phase	XO_ORBIT_ABS_FROM_PHASE_ID	12
xo_orbit_info	XO_ORBIT_INFO_ID	13
xo_osv_to_tle	XO_OSV_TO_TLE_ID	14
xo_run_init	XO_RUN_INIT_ID	15
xo_gen_oef	XO_GEN_OEF_ID	16
xo_gen_osf_create	XO_GEN_OSF_CREATE_ID	17
xo_gen_osf_append_orbit_change	XO_GEN_OSF_APPEND_ORBIT_CHANGE_ID	18
xo_gen_osf_change_repeat_cycle	XO_GEN_OSF_CHANGE_REPEAT_CYCLE_ID	19
xo_gen_osf_add_drift_cycle	XO_GEN_OSF_ADD_DRIFT_CYCLE_ID	20
xo_gen_pof	XO_GEN_POF_ID	21

xo_gen_rof	XO_GEN_ROF_ID	22
xo_gen_rof_prototype	XO_GEN_ROF_PROTOTYPE_ID	23
xo_gen_dnf	XO_GEN_DNF_ID	24
xo_gen_tle	XO_GEN_TLE_ID	25
xo_check_osf	XO_CHECK_OSF_ID	26
xo_check_oef	XO_CHECK_OEF_ID	27
Error Handling Functions		
xo_verbose	not applicable	
xo_silent		
xo_get_code		
xo_get_msg		
xo_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 (XO_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_ORBIT routines, as shown in the table below. The enumerations presented in [GEN_SUM] are also applicable

Table 2: Some enumerations within EO_ORBIT library

Input	Description	Enumeration value	Long
Propagation model	Mean Kepler elements model	XO_PROPAG_MODEL_MEAN_KEPL	0
	SPOT elements model	XO_PROPAG_MODEL_SPOT	1
	TLE model	XO_PROPAG_MODEL_TLE	2
	Precise model (analytical propagator)	XO_PROPAG_MODEL_PRECISE	3
	Auto initialization mode	XO_PROPAG_MODEL_AUTO	10
	Double initialization mode	XO_PROPAG_MODEL_DOUBLE	100
Non Sun-synchronous orbit characterisation	MLST drift	XO_NOSUNSYNC_DRIFT	0
	Inclination	XO_NOSUNSYNC_INCLINATION	1
	Selection of simplified algorithm (additive value)	XO_NOSUNSYNC_USE_SIM_MODEL	10
Time inputs selection	Select the whole file	XO_SEL_FILE	0
	Time	XO_SEL_TIME	1
	Orbit	XO_SEL_ORBIT	2
	Default value	XO_SEL_DEFAULT	3
Interpolation model	Default	XO_INTERPOL_MODEL_DEFAULT	0
Orbit Init Model	Unknown mode	XO_ORBIT_INIT_UNKNOWN_MODE	-1
	Automatic detection of file	XO_ORBIT_INIT_AUTO	0
	Orbit Change mode	XO_ORBIT_INIT_ORBIT_CHANGE_MODE	1
	State Vector mode	XO_ORBIT_INIT_STATE_VECTOR_MODE	2
	Orbit Scenario File mode	XO_ORBIT_INIT_OSF_MODE	3
	Predicted Orbit File mode	XO_ORBIT_INIT_POF_MODE	4
	Restituted Orbit File mode	XO_ORBIT_INIT_ROF_MODE	5
	DORIS mode	XO_ORBIT_INIT_DORIS_MODE	5
	POF refined with DORIS mode	XO_ORBIT_INIT_POF_N_DORIS_MODE	7
	OSF part of the OEF mode	XO_ORBIT_INIT_OEF_OSF_MODE	8
	POF part of the OEF mode	XO_ORBIT_INIT_OEF_POF_MODE	9
	TLE file	XO_ORBIT_INIT_TLE_MODE	10
	State Vector plus precise mode	XO_ORBIT_INIT_STATE_VECTOR_PRECISE_MODE	11
	Predicted Orbit File plus	XO_ORBIT_INIT_POF_PRECISE_MODE	12

	precise mode		
	Restituted Orbit File plus precise mode	XO_ORBIT_INIT_ROF_PRECISE_MODE	13
	DORIS plus precise mode	XO_ORBIT_INIT_DORIS_PRECISE_MODE	14
	Orbit Event File plus precise mode	XO_ORBIT_INIT_OEF_POF_PRECISE_MODE	15
	POF and DORIS files plus precise mode	XO_ORBIT_INIT_POF_N_DORIS_PRECISE_MODE	16
	Maximum value of enumeration	XO_ORBIT_INIT_MAX_VALUE	17
Phase increment	Do not increment phase number at next orbit change	XO_NO_PHASE_INCREMENT	0
	Do increment phase number at next orbit change	XO_PHASE_INCREMENT	1
Orbit change search direction	Search forward	XO_SEARCH_FORWARD	1
	Search backward	XO_SEARCH_BACKWARD	-1
File Type	Orbit Scenario File	XO_REF_FILETYPE_OSF	1
	OSF from an Orbit Event File	XO_REF_FILETYPE_OEF_OSF	2
	FOS Predicted Orbit File	XO_REF_FILETYPE_POF	3
	POF from an Orbit Event File	XO_REF_FILETYPE_OEF_POF	4
	DORIS Navigator File	XO_REF_FILETYPE_DORIS_NAV	5
	FOS Restituted Orbit File	XO_REF_FILETYPE_ROF	6
	DORIS Preliminary Orbit File	XO_REF_FILETYPE_DORIS_PREM	7
	DORIS Precise Orbit File	XO_REF_FILETYPE_DORIS_PREC	8
Precision for ROF and DORIS state vectors times	Default value, non-precise	XO_OSV_PRECISE_NO	1
	Precise location every integer minute	XO_OSV_PRECISE_MINUTE	2
	Precise location every ten seconds	XO_OSV_PRECISE_TEN_SECONDS	3
Number of parameters for Orbit file checking	Number of parameters to check in the functions for checking orbit files	XO_NUM_CHECK_PARAMS	6
TLE generation mode	The requested range of OSV are fitted to one TLE	XO_FIT_TLE	0
	One TLE is generated for every OSV	XO_ONE_TLE_PER_OSV	1
Precise propagator user flag	Use predefined default values for some parameters	XO_DEFAULT_VALUES	0
	Use values introduced by use	XO_USER_VALUES	1
Precise propagator propagation	Do not select contribution	XD_NOT_SELECT	0
	Select contribution	XD_SELECT	1

contribution selection			
Precise propagator propagation SGA input data	Use SGA input parameter	XO_SGA_USE_PARAMETERS	0
	Read SGA values from files	XO_SGA_READ_VALUES_FROM_FILE	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_ORBIT 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_ORBIT structures

Structure name	Data		
	Variable Name	C type	Description
xo_osv_rec	tai_time	double	TAI time for the state vector
	utc_time	double	UTC time for the state vector
	ut1_time	double	UT1 time for the state vector
	abs_orbit	long	Absolute orbit number
	ref_frame	long	Reference frame of the OSV
	pos	double[3]	position of the OSV (x, y, z) components
	vel	double[3]	velocity of the OSV (x, y, z) components
	quality	double	Quality index
xo_anx_extra_info	abs_orbit	long	Absolute orbit number
	tanx	double	ANX time (UT1)
	tnod	double	Nodal period of the orbit
xo_mission_info	abs_orbit	long	Absolute orbit number
	rel_orbit	long	Relative orbit number
	cycle_num	long	Cycle number
	phase_num	long	Phase number
xo_ref_orbit_info	drift_mode	long	Non Sun-synchronous orbit characterisation (see Table 2 for possible values)
	inclination	double	Orbit inclination
	rep_cycle	long	Repeat cycle (days)
	cycle_len	long	Cycle length (orbits)
	ANX_long	double	ANX longitude
	mlst	double	MLST for the ANX
	mlst_drift	double	MLST drift
xo_anx_info	anx_tai	double	TAI time for the ANX
	anx_utc	double	UTC time for the ANX
	anx_ut1	double	UT1 time for the ANX
	anx_pos	double[3]	Position vector
	anx_vel	double[3]	Velocity vector

	kepl	double[6]	Keplerian elements
	tnod	double	Nodal period
xo_osf_records	mission_info	xo_mission_info	Orbit numbers
	ref_orbit_info	xo_ref_orbit_info	Orbit Geometry data
	anx_info	xo_anx_info	ANX Data
xo_validity_time	time_ref	long	Time reference
	start	double	Validity star time
	stop	double	Validity stop time
xo_uni_propag	time_ref	long	Time reference in use
	val_time	xo_validity_time	validity propagation time range in UT1 time
	abs_orbit	long	Predicted Absolute orbit
	time_since_anx	double	Time since ANX
	time	double	Predicted time (UT1)
	pos	double[3]	Osculating position vector at pred. time (EF)
	vel	double[3]	Osculating velocity vector at pred. time (EF)
	acc	double[3]	Osculating acceleration vector at pred. time (EF)
	x	double[6];	Osculating keplerian elements at pred. time (TOD)
xo_propag_id_data	double_propag_flag	long	XL_TRUE if the using double propagation
	accu_mode	long	Flag to indicate if using high or low accuracy mode: 1 = low accuracy 2= high accuracy
	propag_osv	xo_uni_propag	Reference data for propagation
xo_interpol_id_data	time_ref	long	Time reference
	time	double	Time for the interpol reference state vector
	abs_orbit	long	Absolute orbit number
	time_since_anx	double	Time since ANX
	pos	double[3]	Position vector
	vel	double[3]	Velocity vector
	acc	double[3]	Acceleration vector
	kep	double[6]	Keplerian elements
	val_time	xo_validity_time	Interpolation validity time range
xo_propag_precise_	user_flag	long	Indicates if default

config			(XO_DEFAULT_VALUES) or user-defined (XO_USER_VALUES) values are used for some parameters.
	models_path	char[256]	Path where files necessary for models are looked for.
	gravity_flag	long	Gravity perturbation used (XO_SELECT) or not (XO_NOT_SELECT).
	thirdbody_flag	long	Third bodies (Sun and Moon) perturbation used (XO_SELECT) or not (XO_NOT_SELECT).
	atmos_flag	long	Atmosphere perturbation used (XO_SELECT) or not (XO_NOT_SELECT).
	srp_flag	long	Solar radiation pressure perturbation used (XO_SELECT) or not (XO_NOT_SELECT).
	step	double	Simulation step (seconds).
	grav_file	char[256]	File with data of gravitational model.
	grav_degree	long	Degree used gravity model.
	grav_order	long	Order used in gravity model.
	sga_flag	long	ap, f107 and f107a parameters used (XD_SGA_USE_PARAMETERS) or data read from files sga_ap_file and sga_f107_file (XD_SGA_READ_VALUES_FROM_FILE)
	sga_ap_file	char[256]	File with Geomagnetic Activity index values.
	sga_f107_file	char[256]	File with F10.7 Solar Activity index val ues.
	ap	double	Geomagnetic Activity Index (daily value).
	f107	double	F10.7 Index Solar Activity Index (daily value).
	f107a	double	F10.7 Index Solar Activity Index (value averaged over 3 months).
	sc_mass	double	S/C mass [kg].
	sc_drag_area	double	S/C effective drag area [m2].
	sc_drag_coeff	double	S/C drag coefficient.
	sc_srp_area	double	S/C effective Solar Radiation Pressure area [m2].
	sc_srp_coeff	double	S/C Solar Radiation Pressure coefficient

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 xo_orbit_init_def

7.1.1 Overview

The **xo_orbit_init_def** routine generates a Cartesian orbit state vector around the true ascending node crossings. The result is stored and returned through the xo_orbit_id variable so that can fed other routines involving orbit calculations. The data generated by the **xo_orbit_init_def** function is based on:

- Date (processing time),
- Longitude of the ascending node,
- Satellite Repeat Cycle and Cycle Length
- Mean local solar time at ascending node
- Drift of mean local solar time or the inclination

The user should take into account that **xo_orbit_init_def** only retrieve and stores internal data for one orbit.

The validity start and stop times of the initialization (***val_time0*** and ***val_time1*** output parameters) represents the allowed time window for orbit calculations. If the **xo_orbit_init_def** function is called, this time window starts at 01/01/1950 00:00:00 and ends at 31/12/2099 23:59:59.

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EO_LIB functions (see [LIB_SUM]).

Warning: The algorithm used in this function is only valid for satellites with a finite valid range for the inclination and the semi-major axis of the orbit. In CRYOSAT, for example, as there are no minimum and maximum values defined of these two orbital elements, there are defined provisional ranges of the same size as the ones defined in ENVISAT until new requirements are defined. The nominal values have been taken from the [MCD]. There is not available any other nominal orbital element for any other satellite, so this routine is only valid (at this moment) for both CRYOSAT and ENVISAT.

A complete calling sequence of the orbit calculations procedure is presented in section 4.2.

7.1.2 Calling interface

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

```
#include <explorer_orbit.h>
{
      long sat_id, propag_model, time_ref, time_init_mode;
      xl_model_id model_id = {NULL};
      xl_time_id time_id = {NULL};
      xo_orbit_id orbit_id = {NULL};
      long drift_mode, irep, icyc;
      long orbit0, orbit;
      double time0, time, val_time0, val_time1;
      double ascmlst_drift, inclination, rlong, ascmlst;
      long status, ierr[XO_NUM_ERR_ORBIT_INIT_DEF];
      status = xo_orbit_init_def (&sat_id, &model_id, &time_id,
                                  &time_ref, &time0, &orbit0,
                                  &drift_mode,
                                  &ascmlst_drift, &inclination,
                                  &irep, &icyc, &rlong, &ascmlst,
                                  &val_time0, &val_time1,
                                  &orbit_id, ierr);
}
```

7.1.3 Input parameters

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

Table 4: Input parameters of xo_orbit_init_def function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations	-	-
time_ref	long*	-	Time reference ID	-	Complete
time0	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the reference orbit	-	>= 0
drift_mode	long*	-	Flag to select between drift in mean local solar time and inclination as input characterization of the reference orbit. Note: When initializing a Sun-synchronous orbit, the selected drift mode must be *XO_NOSUNSYNC_DRIFT* and the ascmlst_drift parameter must be set to zero. Note 2: Add *XO_NOSUNSYNC_USE_SIM_MODEL* to the drift mode to select the simplified model in the algorithm.	-	XO_NOSUNSYNC_DRIFT, XO_NOSUNSYNC_INCLINATION, XO_NOSUNSYNC_DRIFT + XO_NOSUNSYNC_USE_SIM_MODEL, XO_NOSUNSYNC_INCLINATION + XO_NOSUNSYNC_USE_SIM_MODEL
ascmlst_drift	double*	-	If *drift_mode* = *XO_NOSUNSYNC_DRIFT* Drift in mean local solar time of the reference orbit: · MLST[N+1]=MLST[N]+MLSTdrift See drift_mode entry in this table.	seconds/day	TBD
inclination	double*	-	If *drift_mode* = *XO_NOSUNSYNC_INCLINATION*	deg	[0,180]

			Inclination of the reference orbit		
irep	long *	-	Repeat cycle of the reference orbit The actual repeat cycle is calculated as per definition included in [MCD]	days	> 0
icyc	long *	-	Cycle length of the reference orbit	orbits	> 0
rlong	double*	-	Geocentric longitude of the [Earth fixed] ascending node (Earth fixed CS)	deg	[0,360)
ascmlst	double*	-	Mean local solar time at ascending node	Decimal hours	[0, 24)

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

- Satellite ID: sat_id. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].
- Time initialisation mode: time_init_mode. See [GEN_SUM].
- Drift mode: drift_mode. Current document, section 6.2.

7.1.4 Output parameters

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

Table 5: Output parameters of xo_orbit_init_def function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_init_def	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
orbit_id	xo_orbit _id*	-	Structure that contains the orbit initialization.	-	-
ierr[XO_NUM_ERR_ORBIT_INIT_DEF]	long	all	Status vector	-	-

7.1.5 Warnings and errors

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

Table 6: Error messages of xo_orbit_init_def function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite flag	No calculation performed	XO_CFI_ORBIT_INIT_DEF_SAT_ERR	0
ERR	Wrong input flag: %s	No calculation performed	XO_CFI_ORBIT_INIT_DEF_FLAG_ERR	1
ERR	Could not perform a time transformation	No calculation performed	XO_CFI_ORBIT_INIT_DEF_TIME_CHANGE_ERR	2
ERR	Input out of range: %s	No calculation performed	XO_CFI_ORBIT_INIT_DEF_INPUTS_ERR	3
ERR	An error ocurred in the genstate routine	No calculation performed	XO_CFI_ORBIT_INIT_DEF_GENSTATE_ERR	4
ERR	Memory Error	No calculation performed	XO_CFI_ORBIT_INIT_DEF_MEMORY_ERR	5

7.1.6 Runtime performances

The following runtime performance has been measured.

Table 7: Runtime performances of xo_orbit_init_def function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
1.232	0.316	0.369	0.127

7.2 xo_orbit_cart_init

7.2.1 Overview

This software initializes the orbit data using as input a Cartesian orbit state vector.

The validity start and stop times of the initialization (***val_time0*** and ***val_time1*** output parameters) represents the allowed time window for orbit calculations. If the **xo_orbit_cart_init** function is called, this time window starts at 01/01/1950 00:00:00 and ends at 31/12/2099 23:59:59.

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EO_LIB functions (see [LIB_SUM]).

A complete calling sequence of the orbit calculations procedure is presented in section 4.2.

7.2.2 Calling interface

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

```
#include <explorer_orbit.h>
{
       xl_model_id model_id = {NULL};
       xl_time_id time_id = {NULL};
       xo_orbit_id orbit_id = {NULL};
       long sat_id, time_ref, abs_orbit;
       double time, pos[3], vel[3], val_time0, val_time1;
       long status, ierr[XO_NUM_ERR_ORBIT_CART_INIT];

       status = xo_orbit_cart_init(&sat_id, &model_id, &time_id,
                                   &time_ref, &time,
                                   pos, vel, &abs_orbit,
                                  &val_time0, &val_time1,
                                  &orbit_id, ierr);
}
```

7.2.3 Input parameters

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

Table 8: Input parameters of xo_orbit_cart_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations	-	-
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
pos	double[3]	all	Initial osculating position vector (X, Y, Z) (EF reference frame)	m	-
vel	double[3]	all	Initial osculating velocity vector (X, Y, Z) (EF reference frame)	m/s	-
abs_orbit	long*	-	Orbit of the state vector	-	> 0

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

- Satellite ID: sat_id. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.2.4 Output parameters

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

Table 9: Output parameters of xo_orbit_cart_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_cart_init	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]

orbit_id	xo_orbit_id *	-	Structure that contains the orbit initialization.	-	-
ierr[XO_NUM_ERR_ORBIT_CART_INIT]	long	all	Status vector	-	-

7.2.5 Warnings and errors

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

Table 10: Error messages of xo_orbit_cart_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong Satellite Id.	No calculation performed	XO_CFI_ORBIT_CART_INIT_SAT_ERR	0
ERR	Wrong input flag	No calculation performed	XO_CFI_ORBIT_CART_INIT_FLAG_ERR	1
ERR	Input Time Id. is not initialized.	No calculation performed	XO_CFI_ORBIT_CART_INIT_TIME_STATUS_ERR	2
ERR	Orbit Id is already initialized.	No calculation performed	XO_CFI_ORBIT_CART_INIT_STATUS_ERR	3
ERR	Time conversion error.	No calculation performed	XO_CFI_ORBIT_CART_INIT_TIME_TRANSFORMING_ERR	4
ERR	Time out of limits.	No calculation performed	XO_CFI_ORBIT_CART_INIT_TIME_RANGE_ERR	5
ERR	Memory allocation error.	No calculation performed	XO_CFI_ORBIT_CART_INIT_MEMORY_ERR	6

7.2.6 Runtime performances

The following runtime performance has been measured.

Table 11: Runtime performances of xo_orbit_cart_init function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.166	0.029	0.039	0.0048

7.3 xo_orbit_cart_init_precise

7.3.1 Overview

This software initializes the orbit data using as input a Cartesian orbit state vector for precise propagation (the state vectors will be computed with a numeric propagator).

The validity start and stop times of the initialization (***val_time0*** and ***val_time1*** output parameters) represents the allowed time window for orbit calculations. If the **xo_orbit_cart_init_precise** function is called, this time window starts at the time of the state vector and ends at 31/12/2099 23:59:59.

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EO LIB functions (see [LIB_SUM]).

A complete calling sequence of the orbit calculations procedure is presented in section 4.2.

7.3.2 Calling interface

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

```
#include <explorer_orbit.h>
{
       xl_model_id model_id = {NULL};
       xl_time_id time_id = {NULL};
       xo_orbit_id orbit_id = {NULL};
       long sat_id, time_ref, abs_orbit;
       double time, pos[3], vel[3], val_time0, val_time1;
       xo_propag_precise_config precise_conf;
       long status, ierr[XO_NUM_ERR_ORBIT_CART_INIT];
       status = xo_orbit_cart_init_precise(&sat_id, &model_id,
                                           &time_id,
                                          &time_ref, &time,
                                           pos, vel, &abs_orbit,
                                          &precise_conf,
                                          &val_time0, &val_time1,
                                          &orbit_id, ierr);
}
```

7.3.3 Input parameters

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

Table 12: Input parameters of xo_orbit_cart_init_precise function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations	-	-
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
pos	double[3]	all	Initial osculating position vector (X, Y, Z) (EF reference frame)	m	-
vel	double[3]	all	Initial osculating velocity vector (X, Y, Z) (EF reference frame)	m/s	-
abs_orbit	long*	-	Orbit of the state vector	-	> 0
precise_conf	xo_propag_precise_config*	-	Configuration parameters for precise propagator.	-	struct members with restrictions: - All flags: 0 or 1. - step: > 0. - grav_degree, grav_order > 0. - ap, f107, f107a: >= 0. - sc_mass: > 0. - sc_drag_area: > 0. - sc_drag_coeff: > 0. - sc_srp_area: > 0. - sc_srp_coeff: > 0.

In precise_conf parameter, at least user_flag, models_path and satellite values (sc_mass, sc_drag_area, sc_drag_coeff, sc_srp_area, sc_srp_coeff) must be provided. The other values must be provided just in case the user does not want to use default values (user_flag = XO_USER_VALUES). If default values are seleted (user_flag = XO_DEFAULT_VALUES), then the following values are used:

- gravity_flag = XO_SELECT;
- thirdbody_flag = XO_SELECT;

- atmos_flag = XO_SELECT;
- srp_flag = XO_SELECT;
- step = 10. [s];
- grav_file = egm96.grv;
- grav_degree = 10;
- grav_order = 10;
- sga_flag = XO_SGA_READ_VALUES_FROM_FILE (Use files, not constant values for AP and F107A);
- sga_ap_file = ap_esa_ecss_jan2000_mean.sga;
- sga_f107_file = f107_esa_ecss_jan2000_mean.sga;
- ap = 0.;
- f107 = 0.;
- f107a = 0.;

Some files that can be used or taken as example by the user are provided in the files/models directory of the EOCFI libraries. There are files for gravity model (egm96.grv), F10.7 index (f107_*.sga) and Geomagnetic activity index (ap_*.sga).

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

- Satellite ID: sat_id. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.3.4 Output parameters

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

Table 13: Output parameters of xo_orbit_cart_init_precise function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_cart_init_precise	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	[-18262.0,36524.0]
orbit_id	xo_orbit_id *	-	Structure that contains the orbit initialization.	-	-
ierr[XO_NUM_ERR_ORBIT_CART_INIT_PRECISE]	long	all	Status vector	-	-

7.3.5 Warnings and errors

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

Table 14: Error messages of xo_orbit_cart_init_precise function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error in call to function xo_orbit_cart_init.	No calculation performed	XO_CFI_ORBIT_CART_INIT_FLAG_ERR	0
ERR	Error initialising propagation.	No calculation performed	XO_CFI_ORBIT_CART_INIT_PRECISE_PROPAG_INIT_ERR	1
ERR	Time conversion error.	No calculation performed	XO_CFI_ORBIT_CART_INIT_PRECISE_TIME_CONVERSIO N_ERR	2
ERR	Error in precise propagator input parameters	No calculation performed	XO_CFI_ORBIT_CART_INIT_PRECISE_PRECISE_PARAMS_ERR	3

7.3.6 Runtime performances

The following runtime performance has been measured.

Table 15: Runtime performances of xo_orbit_cart_init_precise function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TBD	TBD	TBD	TBD

7.4 xo_orbit_init_file

7.4.1 Overview

The **xo_orbit_init_file** function is used for initializing the orbit calculations using one of these orbit files:

- One or more FOS Predicted ascending node cartesian state vectors file. In case multiple files are used, the files should be time ordered and the gap between them (i.e. time difference between the last vector of nth file and the first vector of the nth+1 file) should be less than two orbital periods.
- One FOS Predicted Orbit File plus a DORIS Navigator unconsolidated level-0 products file.
- One Orbit Scenario File providing orbital changes.
- One or more Orbit Event files.
- One or more FOS Restituted orbit files.
- One or more DORIS Navigator files.
- One or more DORIS Predicted files.
- One or more DORIS Preliminary files.
- State vectors from Spot orbit files.
- TLE files. In this case it could be neccesary use the function **xl_set_tle_sat_data** before calling **xo_orbit_init_file**.

The format of these files is described in [FORMATS].

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EO LIB functions (see [LIB_SUM]).

In order to prevent incosistent results in computations that will be using the orbit id, the time correlations defined in the orbit file (if any) shall be consistent with those used to initialize the time id.

The user can select the time interval to be used from the input file(s) using three different ways:

Table 16: User requested time range in xo_orbit_init_file

time_mode (see 7.28.4)	input parameter	requested start time (t_req_start)	requested stop time (t_req_stop)
XL_SEL_TIME	time0 / time1	time0	time1
XL_SEL_ORBIT	orbit0 / orbit1	tANX(orbit0)	tANX(orbit1)
XL_SEL_FILE	-	first state vector in the file(s)	last state vector in the file(s)

The validity start and stop times of the initialization (**val_time0** and **val_time1** output parameters) represents the allowed time window for orbit calculation. The following table shows the validity time interval for the different input files:

Table 17: Validity periods for xo_orbit_init_file

Input file type	val_time0	val_time1
Orbit file providing Orbit changes	ANX Time of the first orbital change	Infinity
Orbit files providing a list of orbital state vectors	time of the first state vector	Time of the last state vector
TLE files	time of the first TLE	Time of the last TLE + 1day

A complete calling sequence of the orbit calculation procedure is presented in section 4.2.

7.4.2 Calling interface

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

```
#include <explorer_orbit.h>
{
       xl_time_id time_id = {NULL};
       xl_model_id model_id = {NULL};
       xo_orbit_id orbit_id = {NULL};
       long sat_id, orbit_file_mode, n_files, time_mode;
       long time_ref, orbit0, orbit1;
       char **input_files;
       double time0, time1, val_time0, val_time1;
       long status, ierr[XO_NUM_ERR_ORBIT_INIT_FILE];

       status = xo_orbit_init_file (&sat_id, &model_id, &time_id,
                                    &orbit_file_mode, &n_files,
                                    input_files,
                                    &time_mode, &time_ref,
                                    &time0, &time1, &orbit0, &orbit1,
                                   &val_time0, &val_time1,
                                   &orbit_id, ierr);
}
```

7.4.3 Input parameters

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

Table 18: Input parameters of xo_orbit_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xo_model_id*	-	Model ID	-	-
time_id	xo_time _id*	-	Structure that contains the time correlations	-	-
orbit_file_mode	long*	-	Flag that indicates the type of the input orbit file. • There exists the possibility of detecting automatically the type of the files using the value XO_ORBIT_INIT_AUTO. • The Orbit Event files are used as Orbit Scenario files if the AUTO mode is selected. In case they want to be used as Predicted orbit files, the option XO_ORBIT_INIT_OEF_POF_MODE should be chosen. • The AUTO mode cannot be used to detect TLE files.	-	XO_ORBIT_INIT_AUTO XO_ORBIT_INIT_OSF_ MODE XO_ORBIT_INIT_POF_ MODE XO_ORBIT_INIT_ROF_ MODE XO_ORBIT_INIT_DORIS _MODE XO_ORBIT_INIT_POF_N _DORIS_MODE XO_ORBIT_INIT_OEF_O SF_MODE XO_ORBIT_INIT_OEF_P OF_MODE XO_ORBIT_INIT_TLE_M ODE
n_files	long	-	Number of input files	-	>=1
input_files	char**	-	Vector of orbit files	-	-
time_init_mode	long*	-	Flag for selecting the time range of the initialisation. For TLE files, the whole file is always selected (this flag and the parameters time0/time1, orbit0/orbit1 are dummies)	-	Select either: · XO_SEL_FILE · XO_SEL_ORBIT · XO_SEL_TIME For DORIS Navigator files, XO_SEL_ORBIT is not allowed
time_ref	long*	-	Time reference ID	-	Complete When using DORIS

					Navigator files and time_mode is XO_SEL_TIME, only XL_TIME_UTC is allowed.
time0	double*	-	Start time. See section 7.28.1. Used only if: · time_init_mode=XO_SEL_TIME	Decimal days (Processing format)	[-18262.0,36524.0]
time1	double*	-	Stop time. Used only if: · time_init_mode=XO_SEL_TIME	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the start orbit. Used only if: · time_init_mode=XO_SEL_ORBIT	-	-
orbit1	long*		Absolute orbit number of the stop orbit. Used only if: · time_init_mode=XO_SEL_ORBIT	-	-

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

- Satellite ID: sat_id. See [GEN_SUM].
- Orbit init mode: orbit_init_mode. Current document, section 6.2.
- Time mode: time_init_mode. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.4.4 Output parameters

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

Table 19: Output parameters of xo_orbit_init_file function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_init_file	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	See 7.28.1
val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	See 7.28.1
orbit_id	xo_orbit_id*	-	Structure that contains the orbit initialization data	-	-
ierr[XO_NUM_ER	long	all	Status vector	-	-

R_ORBIT_INIT_FILE]					

7.4.5 Warnings and errors

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

Table 20: Error messages of xo_orbit_init_file function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong satellite flag.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_SAT_ERR	0
ERR	Wrong input flag.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_FLAG_ERR	1
ERR	The Time Id was not initialized.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_TIME_STATUS_ERR	2
ERR	The Orbit Id is already initialized.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_ORBIT_STATUS_ER R	3
ERR	Memory allocation error.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_MEMORY_ERR	4
ERR	Could not detect input files.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_INPUT_FILES_ERR	5
ERR	Error reading OSF.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_WRONG_OSF_FILE_FORMAT_ERR	6
ERR	Wrong time on input.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_TIME_INPUT_INCORR_ERR	7
ERR	Error while processing DORIS file.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_DORIS_INIT_ERR	8
ERR	Time Conversion Error.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_TIME_CONVERSION_ERR	9
ERR	Input time correlations not compatible with input file(s) time correlations.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_WRONG_TIME_CORRELATIONS_ERR	10

ERR	Error reading input files.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_READ_FILES_ERR	11
ERR	No data read within the input range.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_NO_ENOUGH_DATA_ERR	12
ERR	Error while computing ANX data for the state vectors	No calculation performed	XO_CFI_ORBIT_INIT_FILE_INTERPOL_INIT_ANX_ERR	13
ERR	Error computing the orbit number for every state vector	No calculation performed	XO_CFI_ORBIT_INIT_FILE_CALC_ORBIT_ERR	14
WARN	Warnings while computing ANX data	Calculation performed.	XO_CFI_ORBIT_INIT_FILE_INTERPOL_INIT_ANX_WARN	15
WARN	Warnings during DORIS initialization	Calculation performed.	XO_CFI_ORBIT_INIT_FILE_DORIS_INIT_WARN	16
WARN	Warnings while reading the input file list	Calculation performed.	XO_CFI_ORBIT_INIT_FILE_READ_FILES_WARN	17

7.4.6 Runtime performances

The following runtime performances have been measured:

Table 21: Runtime performances of xo_orbit_init_file function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
4.434	2.909	2.159	0.437

7.5 xo_orbit_init_file_precise

7.5.1 Overview

The **xo_orbit_init_file_precise** function is used for initializing the orbit calculations in the same way as **xo_orbit_inti_file**, but in this case precise propagation will be used in state vector propagation. One of these orbit files can be used:

- One or more FOS Predicted ascending node cartesian state vectors file. In case multiple files are used, the files should be time ordered and the gap between them (i.e. time difference between the last vector of nth file and the first vector of the nth+1 file) should be less than two orbital periods.
- One FOS Predicted Orbit File plus a DORIS Navigator unconsolidated level-0 products file.
- One or more FOS Restituted orbit files.
- One or more DORIS Navigator files.
- One or more DORIS Predicted files.
- One or more DORIS Preliminary files.
- State vectors from Spot orbit files.

The format of these files is described in [FORMATS].

Before calling this function it is required to initialise the time correlations, using either **xl_time_ref_init** or **xl_time_ref_init_file** EO LIB functions (see [LIB_SUM]).

In order to prevent incosistent results in computations that will be using the orbit id, the time correlations defined in the orbit file (if any) shall be consistent with those used to initialize the time id.

The user can select the time interval to be used from the input file(s) using three different ways:

Table 22: User requested time range in xo_orbit_init_file_precise

time_mode (see 7.28.4)	input parameter	requested start time (t_req_start)	requested stop time (t_req_stop)
XL_SEL_TIME	time0 / time1	time0	time1
XL_SEL_ORBIT	orbit0 / orbit1	tANX(orbit0)	tANX(orbit1)
XL_SEL_FILE	-	first state vector in the file(s)	last state vector in the file(s)

The validity start and stop times of the initialization (**val_time0** and **val_time1** output parameters) represents the allowed time window for orbit calculation. The following table shows the validity time interval for the different input files:

Table 23: Validity periods for xo_orbit_init_file_precise

Input file type	val_time0	val_time1
Any	time of the first state vector	Infinity

A complete calling sequence of the orbit calculation procedure is presented in section 4.2.

7.5.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xl_model_id time_id = {NULL};
        xl_time_id time_id = {NULL};
        xo_orbit_id orbit_id = {NULL};
        long sat_id, orbit_file_mode, n_files, time_mode;
        long time_ref, orbit0, orbit1;
        char **input_files;
        xo_propag_precise_config precise_conf;
        double time0, time1, val_time0, val_time1;
        long status, ierr[XO_NUM_ERR_ORBIT_INIT_FILE];

        status = xo_orbit_init_file_precise (&sat_id, &model_id,
                                             &time_id,
                                             &orbit_file_mode, &n_files,
                                             input_files,
                                             &time_mode, &time_ref,
                                             &time0, &time1, &orbit0, &orbit1,
                                             &precise_conf,
                                            &val_time0, &val_time1,
                                            &orbit_id, ierr);
}
```

7.5.3 Input parameters

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

Table 24: Input parameters of xo_orbit_init_file_precise function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xo_model_id*	-	Model ID	-	-
time_id	xo_time_id*	-	Structure that contains the time correlations	-	-
orbit_file_mode	long*	-	Flag that indicates the type of the input orbit file. - There exists the possibility of detecting automatically the type of the files using the value XO_ORBIT_INIT_AUTO.	-	XO_ORBIT_INIT_AUTO XO_ORBIT_INIT_POF_PRECISE_MODE XO_ORBIT_INIT_ROF_PRECISE_MODE XO_ORBIT_INIT_DORIS_PRECISE_MODE XO_ORBIT_INIT_OEF_POF_PRECISE_MODE XO_ORBIT_INIT_POF_N_DORIS_PRECISE_MO DE
n_files	long	-	Number of input files	-	>=1
input_files	char**	-	Vector of orbit files	-	-
time_init_mode	long*	-	Flag for selecting the time range of the initialisation. For TLE files, the whole file is always selected (this flag and the parameters time0/time1, orbit0/orbit1 are dummies)	-	Select either: · XO_SEL_FILE · XO_SEL_ORBIT · XO_SEL_TIME For DORIS Navigator files, XO_SEL_ORBIT is not allowed
time_ref	long*	-	Time reference ID	-	Complete

					When using DORIS Navigator files and time_mode is XO_SEL_TIME, only XL_TIME_UTC is allowed.
time0	double*	-	Start time. See section 7.28.1. Used only if: · time_init_mode=XO_SEL_TIME	Decimal days (Processing format)	[-18262.0,36524.0]
time1	double*	-	Stop time. Used only if: · time_init_mode=XO_SEL_TIME	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the start orbit. Used only if: · time_init_mode=XO_SEL_ORBIT	-	-
orbit1	long*		Absolute orbit number of the stop orbit. Used only if: · time_init_mode=XO_SEL_ORBIT	-	-
precise_conf	xo_propag_precise_config*	-	Configuration parameters for precise propagator.	-	-

For precise_conf, the same rules than in xo_orbit_cart_init_precise apply.

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

- Satellite ID: sat_id. See [GEN_SUM].
- Orbit init mode: orbit_init_mode. Current document, section 6.2.
- Time mode: time_init_mode. See [GEN_SUM].
- Time reference ID: time_ref. See [GEN_SUM].

7.5.4 Output parameters

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

Table 25: Output parameters of xo_orbit_init_file_precise function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_init_file_precise	long	-	Main status flag	-	-1, 0, +1
val_time0	double*	-	Validity start time of the initialization	Decimal days (Processing format)	See 7.28.1

val_time1	double*	-	Validity stop time of the initialization	Decimal days (Processing format)	see 7.28.1
orbit_id	xo_orbit_id*	-	Structure that contains the orbit initialization data	-	-
ierr[XO_NUM_ERR_ORBIT_INIT_FILE_PRECISE]	long	all	Status vector	-	-

7.5.5 Warnings and errors

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

Table 26: Error messages of xo_orbit_init_file_precise function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	File type not allowed for precise propagator initialisation	No calculation performed	XO_CFI_ORBIT_INIT_FILE_PRECISE_NOT_ALLOWED_FILE_TYPE_ERR	0
ERR	Error initialising orbit.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_PRECISE_INIT_FILE_ERR	1
ERR	Error initialising propagator.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_PRECISE_PROPAG_INIT_ERR	2
ERR	Could not detect input files.	No calculation performed	XO_CFI_ORBIT_INIT_FILE_PRECISE_INPUT_FILES_ERR	3
ERR	Error in precise propagator input parameters	No calculation performed	XO_CFI_ORBIT_INIT_FILE_PRECISE_PRECISE_PARAMS_ERR	4

7.5.6 Runtime performances

The following runtime performances have been measured:

Table 27: Runtime performances of xo_orbit_init_file_precise function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TBD	TBD	TBD	TBD

7.6 xo_orbit_close

7.6.1 Overview

The **xo_orbit_close** function is used to free the memory allocated by the other orbit initialization routines, and it must be called after using them.

A complete calling sequence of the propagation procedure is presented in section 4.2.

7.6.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id = {NULL};
        long ierr[XO_NUM_ERR_ORBIT_CLOSE]
        long status;

        status = xo_orbit_close (&orbit_id, ierr);
}
```

7.6.3 Input parameters

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

Table 28: Input parameters of xo_orbit_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id *	-	Structure that contains the orbit initialization	-	-

7.6.4 Output parameters

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

Table 29: Output parameters of xo_orbit_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
ierr[XO_NUM_ERR_ORBIT_CLOSE]	long	all	Status vector	-	-
xo_orbit_close	long	-	Main status flag	-	-1, 0, +1

7.6.5 Warnings and errors

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

Table 30: Error messages of xo_orbit_close function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Could not close the Orbit Id.	The Orbit Id. was not closed.	XO_CFI_ORBIT_CLOSE_ WRONG_ID_ERR	0

7.6.6 Runtime performances

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

7.7 xo_orbit_get_osv

7.7.1 Overview

The **xo_orbit_get_osv** CFI function returns a data structure containing the list of state vectors used for the initialisation of an orbit_id. This function only can be called if the orbit_id was initialized with orbital state vectors (i.e., with **xo_orbit_cart_init** or with **xo_orbit_init_file** and a file containing a list of state vectors such as predicted orbit file, a restituted orbit file...)

7.7.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long num_rec;
        xo_osv_rec* data;
        long status;
        status = xo_orbit_get_osv(&orbit_id, &num_rec, &data);
}
```

7.7.3 Input parameters

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

Table 31: Input parameters of xo_orbit_get_osv function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_id	xo_orbit_id*	-	Structure for orbit initiaization	-	-

7.7.4 Output parameters

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

Table 32: Output parameters of xo_orbit_get_osv function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_osv	long	-	Status flag	-	-
num_rec	long	-	Number of records in the data array	-	-

data	xo_osv_rec	all	Dinamic array with the state vectors	-	-

The data structure xo_osv_rec can be seen in Table 3.

Note: The output *data* array is a pointer, not a static array. The memory for this dynamic array is allocated within the CFI function. So the user will only have to declare that pointer but not to allocate memory for it. However, once the function has returned without error, the user will have the responsibility of freeing the memory when it is not being used any more. For freeing the memory just call to (in a C program):

```
free(data);
```

7.7.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.
- The orbit_id was initialised with orbital changes, instead of state vectors.

7.7.6 Runtime performances

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

7.8 xo_orbit_set_osv

7.8.1 Overview

The **xo_orbit_set_osv** CFI function changes the list of state vectors used for the initialisation within an orbit_id. This function only can be called if the orbit_id was initialized with orbital state vectors (i.e., with **xo_orbit_cart_init** or with **xo_orbit_init_file** and a file containing a list of state vectors such as predicted orbit file, a restituted orbit file...)

7.8.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long num_rec;
        xo_osv_rec* data;
        long status;
        status =    xo_orbit_set_osv(&orbit_id, &num_rec, data);
}
```

7.8.3 Input parameters

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

Table 33: Input parameters of xo_orbit_set_osv function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-
num_rec	long	-	Number of records in the data array	-	-
data	xo_osv_rec	all	Dynamic array with the state vectors	-	-

7.8.4 Output parameters

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

Table 34: Output parameters of xo_orbit_set_osv function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_set_osv	long	-	Status flag	-	-
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-

7.8.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.
- The orbit_id was initialised with orbital changes, instead of state vectors.

7.8.6 Runtime performances

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

7.9 xo_orbit_get_anx

7.9.1 Overview

When initialising an orbit_id with a list of state vectors that are not in the ANX (restituted orbit file, DORIS Navigator files), the information about the ANX of the orbits of those state vectors are stored in the orbit_id. The **xo_orbit_get_anx** CFI function allows to retrieve that information.

This function only can be called if the orbit_id was initialized with orbital state vectors coming from:

- Restituted orbit file
- DORIS Navigator file

7.9.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long num_rec;
        xo_anx_extra_info* extra_info;
        long status;
        status = xo_orbit_get_anx(&orbit_id, &num_rec, &extra_info);
}
```

7.9.3 Input parameters

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

Table 35: Input parameters of xo_orbit_get_anx function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.9.4 Output parameters

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

Table 36: Output parameters of xo_orbit_get_anx function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_anx	long	-	Status flag	-	-
num_rec	long	-	Number of records in the data array	-	-
extra_info	xo_anx_extra_info	all	Dinamic array with the ANX information	-	-

The data structure xo_osv_rec can be seen in Table 3.

Note: The output *extra_info* array is a pointer, not a static array. The memory for this dynamic array is allocated within the CFI function. So the user will only have to declare that pointer but not to allocate memory for it. However, once the function has returned without error, the user will have the responsibility of freeing the memory when it is not being used any more. For freeing the memory just call to (in a C program):

```
free(extra_info);
```

7.9.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.
- The orbit_id was not initialised with the suitable file

7.9.6 Runtime performances

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

7.10 xo_orbit_set_anx

7.10.1 Overview

The **xo_orbit_set_anx** CFI function changes the ANX info that is stored in an orbit_id when this orbit id was initialised with a restituted orbit file or a DORIS Navigator file.

7.10.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long num_rec;
        xo_anx_extra_info* extra_info;
        long status;
        status =    xo_orbit_set_anx(&orbit_id, &num_rec, extra_info);
}
```

7.10.3 Input parameters

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

Table 37: Input parameters of xo_orbit_set_anx function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-
num_rec	long	-	Number of records in the data array	-	-
extra_info	xo_anx_extra_info	all	Dynamic array with the state vectors	-	-

7.10.4 Output parameters

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

Table 38: Output parameters of xo_orbit_set_anx function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_set_anx	long	-	Status flag	-	-
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-

7.10.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.
- The orbit_id was initialised with orbital changes, instead of state vectors.

7.10.6 Runtime performances

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

7.11 xo_orbit_get_osf_rec

7.11.1 Overview

The **xo_orbit_get_osf_rec** CFI function returns a data structure containing the list of orbital changes used for the initialisation of an orbit_id. This function only can be called if the orbit_id was initialized with orbital changes(i.e., with **xo_orbit_init_def** or with **xo_orbit_init_file** and an orbit scenario file)

7.11.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long num_rec;
        xo_osf_records* data;
        long status;
        status =    xo_orbit_get_osf_rec(&orbit_id, &num_rec, &data);
}
```

7.11.3 Input parameters

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

Table 39: Input parameters of xo_orbit_get_osf_rec function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.11.4 Output parameters

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

Table 40: Output parameters of xo_orbit_get_osf_rec function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_osf_rec	long	-	Status flag	-	-
num_rec	long	-	Number of records in the data array	-	-
data	xo_osf_rec	all	Dinamic array with	-	-

			the orbital changes		

The data structure xo_osf_rec can be seen in Table 3.

Note: The output *data* array is a pointer, not a static array. The memory for this dynamic array is allocated within the CFI function. So the user will only have to declare that pointer but not to allocate memory for it. However, once the function has returned without error, the user will have the responsibility of freeing the memory when it is not being used any more. For freeing the memory just call to (in a C program):

```
free(data);
```

7.11.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.
- The orbit_id was not initialised with orbital changes.

7.11.6 Runtime performances

The following runtime performances have been estimated.

Table 41: Runtime performances of xo_orbit_get_osf_rec function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.0018	0.0008	0.0016	0.0002

7.12 xo_orbit_set_osf_rec

7.12.1 Overview

The **xo_orbit_set_osf_rec** CFI function changes the list of orbital changes used for the initialisation within an orbit_id. This function only can be called if the orbit_id was initialized with **xo_orbit_init_def** or with **xo_orbit_init_file** and an orbit scenario file.

7.12.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long num_rec;
        xo_osv_rec* data;
        long status;
        status =    xo_orbit_set_osf_rec(&orbit_id, &num_rec, data);
}
```

7.12.3 Input parameters

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

Table 42: Input parameters of xo_orbit_set_osf_rec function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-
num_rec	long	-	Number of records in the data array	-	-
data	xo_osf_rec	all	Dinamic array with the orbital changes	-	-

7.12.4 Output parameters

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

Table 43: Output parameters of xo_orbit_set_osf_rec function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_set_osf_rec	long	-	Status flag	-	-
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-

7.12.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.
- The orbit_id was not initialised with orbital changes.

7.12.6 Runtime performances

The following runtime performances have been estimated.

Table 44: Runtime performances of xo_orbit_set_osf_rec function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.0048	0.0018	0.0016	0.0002

7.13 xo_orbit_get_val_time

7.13.1 Overview

The **xo_orbit_get_val_time** CFI function returns the validity period of an orbit_id.

7.13.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        xo_validity_time val_time;
        long status;
        status =    xo_orbit_get_val_time(&orbit_id, &val_time);
}
```

7.13.3 Input parameters

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

Table 45: Input parameters of xo_orbit_get_val_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.13.4 Output parameters

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

Table 46: Output parameters of xo_orbit_get_val_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_val_time	long	-	Status flag	-	-
val_time	xo_validity_time	-	Validity Time structure	-	-

The data structure xo_validity_time can be seen in Table 3.

7.13.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.

7.13.6 Runtime performances

The following runtime performances have been estimated: runtime is smaller than CPU clock and it is not possible to perform loops for measuring it.

7.14 xo_orbit_set_val_time

7.14.1 Overview

The **xo_orbit_set_val_time** CFI function changes the validity period of an orbit_id.

7.14.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        xo_validity_time val_time;
        long status;
        status =    xo_orbit_set_val_time(&orbit_id, &val_time);
}
```

7.14.3 Input parameters

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

Table 47: Input parameters of xo_orbit_set_val_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-
val_time	xo_validity_time	-	Validity Time structure	-	-

7.14.4 Output parameters

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

Table 48: Output parameters of xo_orbit_set_val_time function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_set_val_time	long	-	Status flag	-	-
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-

7.14.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.

7.14.6 Runtime performances

The following runtime performances have been estimated.

Table 49: Runtime performances of xo_orbit_set_val_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.0002	0.0002	0	0

7.15 xo_orbit_get_precise_propag_config

7.15.1 Overview

The **xo_orbit_get_precise_propag_config** CFI function returns the configuration structure of precise propagation.

7.15.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        xo_propag_precise_config precise_conf;
        long status;
        status =    xo_orbit_get_precise_propag_config(&orbit_id,
                                                       &precise_conf);
}
```

7.15.3 Input parameters

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

Table 50: Input parameters of xo_orbit_get_precise_propag_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-

7.15.4 Output parameters

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

Table 51: Output parameters of xo_orbit_get_precise_propag_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_precise_propag_config	long	-	Status flag	-	-
precise_conf	xo_propag_precise_config	-	Precise propagator configuration	-	-

			structure		

7.15.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.

7.15.6 Runtime performances

The following runtime performances have been estimated.

Table 52: Runtime performances of xo_orbit_get_precise_propag_config function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]

7.16 xo_orbit_set_precise_propag_config

7.16.1 Overview

The **xo_orbit_set_precise_propag_config** CFI function sets the configuration structure of precise propagation.

7.16.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        xo_propag_precise_config precise_conf;
        long status;
        status =    xo_orbit_set_precise_propag_config(&orbit_id,
                                                       &precise_conf);
}
```

7.16.3 Input parameters

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

Table 53: Input parameters of xo_orbit_set_precse_propag_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initial-ization (input / output parameter)	-	-
precise_conf	xo_propag_precise_config	-	Precise propagator configuration structure	-	-

7.16.4 Output parameters

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

Table 54: Output parameters of xo_orbit_set_precse_propag_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_set_precise_propag_config	long	-	Status flag	-	-

orbit_id	xo_orbit_id*	-	Structure for orbit initialization (input / output parameter)	-	-

7.16.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.

7.16.6 Runtime performances

The following runtime performances have been estimated.

Table 55: Runtime performances of xo_orbit_set_precse_propag_config function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]

7.17 xo_orbit_get_time_id

7.17.1 Overview

The **xo_orbit_get_time_id** CFI function returns the time_id structure used for the orbit_id initialisation.

7.17.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        xl_time_id time_id;
        time_id =   xo_orbit_get_time_id(&orbit_id);
}
```

7.17.3 Input parameters

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

Table 56: Input parameters of xo_orbit_get_time_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.17.4 Output parameters

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

Table 57: Output parameters of xo_orbit_get_time_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_time_id	xl_time_id	-	time id used for the orbit_id initialisation	-	-

7.17.5 Warnings and errors

This function does not return any error/warning code. In case of error, an empty time_id is returned (initialised with NULL)

The possible causes of error are:

- The orbit_id was not initialised

7.17.6 Runtime performances

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

7.18 xo_orbit_get_model_id

7.18.1 Overview

The **xo_orbit_get_model_id** CFI function returns the model_id structure used for the orbit_id initialisation.

7.18.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id = {NULL};
        xl_model_id model_id = {NULL};
        model_id =  xo_orbit_get_model_id(&orbit_id);
}
```

7.18.3 Input parameters

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

Table 58: Input parameters of xo_orbit_get_model_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.18.4 Output parameters

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

Table 59: Output parameters of xo_orbit_get_model_id function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_model_id	xl_model_id	-	model id used for the orbit_id initialisation	-	-

7.18.5 Warnings and errors

This function does not return any error/warning code. In case of error, an empty time_id is returned (initialised with NULL)

The possible causes of error are:

- The orbit_id was not initialised.

7.18.6 Runtime performances

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

7.19 xo_orbit_get_osv_compute_validity

7.19.1 Overview

The **xo_orbit_get_osv_compute_validity** CFI function returns the validity time interval where it is possible to compute an state vector using the CFI function **xo_osv_compute**. Out of this interval, the functions would return an error.

The validity interval for using **xo_osv_compute** depends on the type of data used for the orbit initialisation. In general, that interval will be different from the validity of the input orbit_id. More information about the validity interval for **xo_osv_compute** can be found in section 7.28.2.

7.19.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        xo_validity_time val_time;
        time_id =   xo_orbit_get_osv_compute_validity(&orbit_id,
                                                      &val_time);
}
```

7.19.3 Input parameters

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

Table 60: Input parameters of xo_orbit_get_osv_compute_validity function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.19.4 Output parameters

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

Table 61: Output parameters of xo_orbit_get_osv_compute_validity function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_osv_compute_validity	xo_validity_time	-	validity time interval for the function **xo_osv_compute**	-	-

The data structure xo_validity_time can be seen in Table 3.

7.19.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.

7.19.6 Runtime performances

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

7.20 xo_orbit_get_propag_mode

7.20.1 Overview

The **xo_orbit_get_propag_mode** CFI function returns the propagation mode that will be used to propagate the state vector when using the input orbit_id.

7.20.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long mode;
        mode =  xo_orbit_get_propag_mode(&orbit_id);
}
```

7.20.3 Input parameters

The **xo_orbit_get_propag_mode** CFI function has the following input parameters:=

Table 62: Input parameters of xo_orbit_get_propag_mode function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.20.4 Output parameters

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

Table 63: Output parameters of xo_orbit_get_propag_mode function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_propag_mode	long	-	propagation mode. -1 if the orbit_id is not initialised for propagation.	-	-

7.20.5 Warnings and errors

This function does not return any error/warning code. If the orbit_id is not initialised or it is not initialised with propagation data, then the returned mode is -1.

7.20.6 Runtime performances

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

7.21 xo_orbit_get_interpol_mode

7.21.1 Overview

The **xo_orbit_get_interpol_mode** CFI function returns the interpolation mode that will be used to interpolate the state vector when using the input orbit_id.

7.21.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long mode;
        mode = xo_orbit_get_interpol_mode(&orbit_id);
}
```

7.21.3 Input parameters

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

Table 64: Input parameters of xo_orbit_get_interpol_mode function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.21.4 Output parameters

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

Table 65: Output parameters of xo_orbit_get_interpol_mode function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_interpol_mode	long	-	propagation mode. -1 if the orbit_id is not initialised for interpolations.	-	-

7.21.5 Warnings and errors

This function does not return any error/warning code. If the orbit_id is not initialised or it is not initialised with interpolation data, then the returned mode is -1.

7.21.6 Runtime performances

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

7.22 xo_orbit_get_propag_config

7.22.1 Overview

The **xo_orbit_get_propag_config** CFI function returns the propagation data that will be used to propagate the state vector when using the input orbit_id.

7.22.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long status;
        xo_propag_id_data propag_data;
        status =    xo_orbit_get_propag_config(&orbit_id,
                                               &propag_data);
}
```

7.22.3 Input parameters

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

Table 66: Input parameters of xo_orbit_get_propag_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.22.4 Output parameters

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

Table 67: Output parameters of xo_orbit_get_propag_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_propag_config	long	-	status	-	-
propag_data	xo_propag_id_data	-	Configuration data used to launch the propagation	-	-

The data structure xo_propag_id_data can be seen in Table 3.

7.22.5 Warnings and errors

This function does not return any error/warning code. If the orbit_id is not initialised or it is not initialised with propagation data, then the returned status is -1.

7.22.6 Runtime performances

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

7.23 xo_orbit_get_interpol_config

7.23.1 Overview

The **xo_orbit_get_interpol_config** CFI function returns the propagation data that will be used to interpoloate the state vector when using the input orbit_id.

7.23.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id;
        long status;
        xo_interpol_id_data interpol_data;
        status =    xo_orbit_get_interpol_config(&orbit_id,
                                                 &interpol_data);
}
```

7.23.3 Input parameters

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

Table 68: Input parameters of xo_orbit_get_interpol_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.23.4 Output parameters

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

Table 69: Output parameters of xo_orbit_get_interpol_config function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_get_propag_config	long	-	status	-	-
interpol_data	xo_interpol_id_data	-	Configuration data used to launch the interpolation	-	-

The data structure xo_interpol_id_data can be seen in Table 3.

7.23.5 Warnings and errors

This function does not return any error/warning code. If the orbit_id is not initialised or it is not initialised with propagation data, then the returned status is -1.

7.23.6 Runtime performances

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

7.24 xo_orbit_id_clone

7.24.1 Overview

The **xo_orbit_id_clone** CFI function copies the input orbit_id structure to the output one.

7.24.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id_in, xo_orbit_id_out;
        long status;
        status =    xo_orbit_id_clone(&orbit_id_in, &orbit_id_out);
}
```

7.24.3 Input parameters

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

Table 70: Input parameters of xo_orbit_id_clone function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id_in	xo_orbit_id*	-	Structure for orbit initialization	-	-

7.24.4 Output parameters

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

Table 71: Output parameters of xo_orbit_id_clone function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_id_clone	long	-	status	-	-
orbit_id_out	xo_orbit_id	-	Output orbit_id	-	-

7.24.5 Warnings and errors

This function does not return any error/warning code. Only the status of the function indicates if the execution was correct or not.

The possible causes of error are:

- The orbit_id was not initialised.

7.24.6 Runtime performances

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

7.25 xo_run_init

7.25.1 Overview

The **xo_run_init** CFI function adds to the *run Id* the *orbit id.*

7.25.2 Calling interface

The calling interface of the **xo_run_init** CFI function is the following:

```
#include <explorer_orbit.h>
{
        long run_id;
        xo_orbit_id orbit_id = {NULL};
        long ierr[XO_NUM_ERR_RUN_INIT], status;
        status =     xo_run_init (&run_id, &orbit_id,
                                  ierr);
}
```

7.25.3 Input parameters

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

Table 72: Input parameters of xo_run_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-

7.25.4 Output parameters

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

Table 73: Output parameters of xo_run_init function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_run_init	long	-	Status flag	-	-
run_id	long *	-	Run ID	-	>=0
ierr	long	-	Error vector	-	-

7.25.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_run_init** CFI function after translating the returned extended status flag into the equivalent list of error messages by calling the function of the EO_ORBIT software library **xo_get_msg** (see [GEN_SUM]).

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

The table is completed by the error code and value. These error codes can be obtained translating the extended status flag returned by the **xo_run_init** function by calling the function of the EO_ORBIT software library **xo_get_code** (see [GEN_SUM])

Table 74: Error messages of xo_run_init function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Inputs Id no initialized or incompatible.	No calculation performed	XO_CFI_RUN_INIT_STATUS_ERR	0
ERR	Memory allocation error.	No calculation performed	XO_CFI_RUN_INIT_MEMORY_ERR	1
ERR	Input Ids incompatible with the run_id.	No calculation performed	XO_CFI_RUN_INIT_INCONSISTENCY_ERR	2

7.25.6 Runtime performances

The following runtime performances have been estimated: runtime is smaller than CPU clock and it is not possible to perform loops for measuring it.

7.26 xo_run_get_ids

7.26.1 Overview

The **xo_run_get_ids** CFI function returns the *ids* being used..

7.26.2 Calling interface

The calling interface of the **xo_run_get_ids** CFI function is the following:

```
#include <explorer_orbit.h>
{
        long run_id;
        xo_orbit_id orbit_id = {NULL};
        xo_run_get_ids (&run_id,
                        &orbit_id);
}
```

7.26.3 Input parameters

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

Table 75: Input parameters of xo_run_get_ids function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0

7.26.4 Output parameters

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

Table 76: Output parameters of xo_run_get_ids function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_run_get_ids	void	-	-	-	-
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-

7.26.5 Warnings and errors

This function does not return any error/warning code.

7.26.6 Runtime performances

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

7.27 xo_run_close

7.27.1 Overview

The **xo_run_close** CFI function cleans up any memory allocation performed by the initialization functions.

7.27.2 Calling interface

The calling interface of the **xo_run_close** CFI function is the following:

```
#include <explorer_orbit.h>
{
        long run_id;
        xo_run_close (&run_id);
}
```

7.27.3 Input parameters

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

Table 77: Input parameters of xo_run_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
run_id	long *	-	Run ID	-	>=0

7.27.4 Output parameters

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

Table 78: Output parameters of xo_run_close function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_run_close	void	-	-	-	-

7.27.5 Warnings and errors

This function does not return any error/warning code.

7.27.6 Runtime performances

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

7.28 xo_osv_compute

7.28.1 Overview

The **xo_osv_compute** function computes accurate state vectors for user requested times.

This function needs as input an orbit_id that contains the orbit initialisation data. The behavior of the function depends strongly in the way in which the orbit_id was initialised (see section 7.28.2).

During the computation, some results are stored in the input orbit_id. Those results can be used afterwards for other computations (in **xo_osv_compute** or **xo_osv_compute_extra**). This has to be taken into account when working with **multi-threads programs**. In those cases a different orbit_id has to be used in every thread to avoid problems. The orbit id can be initialised in every thread or copied (this is more efficient) to different orbit_id variables with **xo_orbit_id_clone**.

An OSV can be computed within a validity time range that also depends on input orbit_id variable. This validity time range can be retrieved with the CFI function **xo_orbit_get_osv_compute_validty** (see section 7.19).

For a general description of the initialization routines and how to use them in conjunction to the **xo_osv_compute** function, see section 4.1.2.

7.28.2 Computation methods (Propagation/interpolation)

The methods used to compute the state vector depends on the way in which the orbit was initialised. Note that the computation method is selected automatically by the routine. The Table 79 summarises the dependency of the computation method with the orbit initialisation. The first column indicates the orbit mode which is a parameter that can be get with the CFI function **xo_orbit_get_mode**. The methods in the second column are described in detail in the following subsections.

Table 79: OSV computation methods

Orbit initialisation mode	OSV computation method
XO_ORBIT_INIT_ORBIT_CHANGE_MODE Initialised with **xo_orbit_init_def**	Mean Keplerian Propagation Model with "AUTO" and "DOUBLE" modes (see section 7.28.2.1 and Table 80)
XO_ORBIT_INIT_OSF_MODE XO_ORBIT_INIT_OEF_OSF_MODE Initialised with **xo_orbit_init_file** with an OSF (or an OEF but reading the list of orbital changes in the data block of the file)	
XO_ORBIT_INIT_POF_MODE XO_ORBIT_INIT_OEF_POF_MODE Initialised with **xo_orbit_init_file** with a POF (or an OEF but reading the list of State Vectors in the data	

block of the file)	
XO_ORBIT_INIT_STATE_VECTOR_MODE Initialised with **xo_orbit_cart_init**	Mean Keplerian Propagation Model (see section 7.28.2.1 and Table 80)
XO_ORBIT_INIT_POF_N_DORIS_MODE: Initialised with **xo_orbit_init_file** with a POF and a DORIS file	
XO_ORBIT_INIT_TLE_MODE Initialised with **xo_orbit_init_file** with a TLE file	TLE propagation mode with "AUTO" mode (see section 7.28.2.1 and Table 80)
XO_ORBIT_INIT_POF_PRECISE_MODE XO_ORBIT_INIT_OEF_POF_PRECISE_MODE Initialised with **xo_orbit_init_file_precise** with a POF (or an OEF but reading the list of State Vectors in the data block of the file)	Numerical propagator with "AUTO" mode (see section 7.28.2.1 and Table 80)
XO_ORBIT_INIT_ROF_PRECISE_MODE Initialised with **xo_orbit_init_file_precise** with a ROF file	
XO_ORBIT_INIT_DORIS_PRECISE_MODE Initialised with **xo_orbit_init_file_precise** with a DORIS Navigator file	
XO_ORBIT_INIT_STATE_VECTOR_PRECISE_MODE Initialised with **xo_orbit_cart_init_precise**	Numerical propagator (see section 7.28.2.1 and Table 80)
XO_ORBIT_INIT_POF_N_DORIS_PRECISE_MODE Initialised with **xo_orbit_init_file_precise** with a POF and a DORIS Navigator file	
XO_ORBIT_INIT_ROF_MODE Initialised with **xo_orbit_init_file** with a ROF file	Interpolation (see section 7.28.2.2)
XO_ORBIT_INIT_DORIS_MODE Initialised with **xo_orbit_init_file** with a DORIS Navigator file	

7.28.2.1 Propagation methods

For the time being, the following propagation models are supported:

- ***Mean Keplerian model***. It implies the use of a formulation for the time rates of change for the different mean Kepler elements as functions of a given initial set of mean Kepler elements. Using the above time rates of change, the mean orbital elements can be propagated forward or backward in time by extrapolating the individual time slopes of the superimposed secular and long-periodic perturbations functions. As the long periodic variations have typically periods on the order of months, a near-linear time slope for prediction intervals of many orbits is warranted.
- ***TLE model***. This model propagates the state vector using the NORAD "two line elements" (TLE) and the SGP4 propagation theroy. This theory was designed for near Earth Satellites (nodal period

less than 225 minutes). The SGP4 theory uses an Earth gravitational field through zonal terms J2, J3 and J4 and a power density function for the atmospheric model (assuming a non-rotating spherical model).

- **Numerical propagator**. This model consists on a numerical propagator that integrates the movement equations using a Runge-Kutta algorithm of 8th order. This propagator is expected to produce more precise results than the other models as it can be configured by the user (through the orbit initialisation function) to take into account the following perturbations:
  - Non-spherical gravity: the model Earth Gravity Model 1996 is used. The file with the coefficients of the spherical harmonics must be provided by the user, besides the order and degree to be used in the calculations.
  - Atmospheric drag: MSIS-E-90 atmospheric density model is used. The user must provide the Solar Geomagnetic Activity and F107 coefficient, either as input files or with constant values. The values can be obtained from ESA-ECSS or NASA documentation. The user must provide the drag effective area and drag coefficient of the satellite, besides the mass of the satellite.
  - Solar radiation pressure: it is calculated using the solar radiation pressure effective area and solar radiation pressure coefficient of the satellite provided by the user.
  - Third body perturbations: the perturbations produced by the Sun and the Moon gravities are calculated.
- ***Spot elements model*** (still TBD). This model is based upon the usage of an extended orbit state vector (originally used for SPOT satellites and currently for MetOp). The calculation of the orbit state vector is made by fitting them using a predicted or restituted orbit file.

Apart from these models, in table there are two additional modes: “AUTO” and “DOUBLE”. They refer to the way in which the seed (initial state vector used as reference to begin the propagation) is taken:

- **AUTO mode**: The seed is taken to be the closer ANX or OSV to the requested time. The propagation seed could change from one propagation to the following.
- **DOUBLE mode:** the two ANX covering the propagation time are used as seeds. When calling **xo_osv_compute**, the propagation is performed from each of the ANX and then a weighted average is done. The weight function is

$$\cos^2\left(\frac{\pi}{2}\cdot\frac{\Delta t}{T}\right)$$

where $\Delta t = t - t_{ANX}$ and T is the nodal period of the orbit.

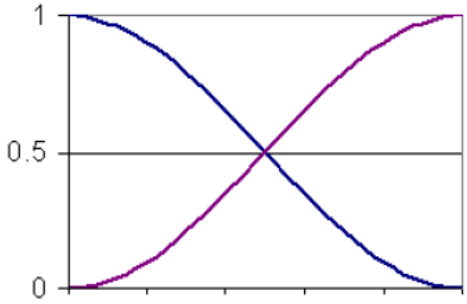
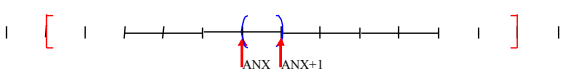
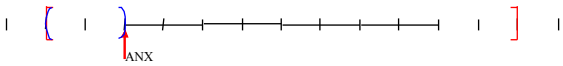
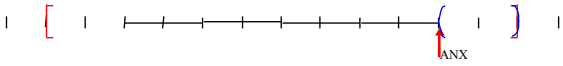
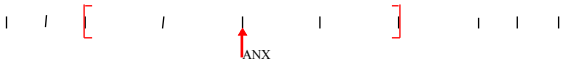
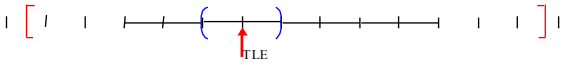


Figure 3: Weight Function for Double Propagation Model

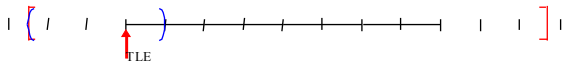
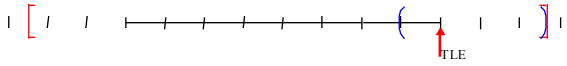
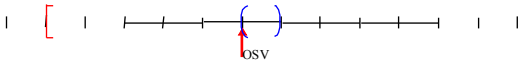
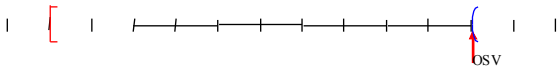
This propagation method removes any discontinuity that may arise when changing the state vector around the true ascending node crossing used as seed for the propagation.

Finally, the validity time interval for the propagation also depends on the propagation method. This validity interval can be computed with the CFI function **xo_orbit_get_osv_compute_validity** (section 7.19). The table summarises all the propagation cases, explaining the seed that is taken in every case and how the validity interval is taken. In the table, $\mathbf{t_0}$ represent the input time for **xo_osv_compute**. In the fourth column, the horizontal solid line in the graphic represents the list of OSV or ANX that are stored in the orbit initialisation, t_start and t_stop are the first and last OSV(ANX) in that line. Red square brackets represent the validity period for propagation. When using the AUTO mode, the seed for the propagation changes if t_0 jumps out of the region in blue brackets. The red arrow(s) represent the chosen seed(s) (OSV or ANX).

Table 80: Validity Time Intervals for Propagation

Propag model[2]	Requested time	Propagation seed	Validity time interval
Mean Keplerian + AUTO + DOUBLE Mode	t_start < t0 < t_stop	The two ANX that are before and after **t0**	[t_start - 2 orbits, t_stop + 2 orbits]; (ANX, ANX + 1orbit) 
	t0 < t_start	The first ANX in the orbit_id	[t_start - 2 orbits, t_stop + 2 orbits];  (ANX - 2 orbits, ANX)
	t0 > t_stop	The last ANX in the orbit_id	[t_start - 2 orbits, t_stop + 2 orbits];  (ANX , ANX + 2 orbits)
Mean Keplerian Mode	t_start - 2orbits < t0 < t_stop+2orbits Note that t_start = t_stop as there is only one OSV	The state vector used to initialise the orbit_id (there is only one)	[t_start - 2 orbits, t_stop + 2 orbits] 
TLE + AUTO Mode	t_start < t0 < t_stop	TLE that is before **t0**	[1st.TLE - 1day, Last TLE+ 1 day] (-1 TLE, +1 TLE) 
	t0 < t_start	First TLE	[1st.TLE - 1day, Last TLE+ 1 day] (TLE - 1day, + 1 TLE)

2 This column relates with the "OSV computation method" column in Table 79.

			
	t0 > t_stop	Last TLE	[1st.TLE - 1day, Last TLE+ 1 day] (-1 TLE, TLE + 1 day) 
Numerical + AUTO Mode	t_start < t0 < t_stop	OSV that is before **t0**	[t_start, Infinity]; (OSV, Next OSV) 
	t0 < t_start	Propagation is not possible	-
	t0 > t_stop	Last OSV	[t_start, Infinity]; (OSV, Infinity) 

Note that the cases with “NO AUTO” mode are similar to those with “AUTO” mode with one OSV.

The precise propagator stores the result of the last propagation, so that the next propagation begins from that point if the time begins after the last propagation (it saves computation time). If the propagation is requested at a time that is before the time of the previous propagation time, then the original seed is taken.

7.28.2.2 Interpolation methods

The function **xo_osv_compute** computes the OSV using an interpolation algorithm when the orbit_id is initialised with **xo_orbit_init_file** plus ROF’s or DORIS Navigator files.

The interpolation is highly accurate (1 mm. accuracy TBC) when it is performed between 4 OSV’s time intervals after start of file(s) and before end of file(s), but it degrades (up to a few cm. TBC) until 1 or 2 time intervals (TBD) before start of file(s) and after end of file(s). Figure 4 provides a graphical explanation.

The **xo_osv_compute** function allows to extrapolate, that is, compute results for the 1 or 2 (TBC) intervals before start of the input file(s) and after enf of the input file. Anyway, extrapolation is not recommended. In this case, the extrapolation window is NOT included in the valid time interval.

When the interpolation is in “degraded” mode, that is, when extrapolation is used, or when there is less than four orbit state vectors available in the input file before or after the requested time, **xo_osv_compute** function will issue different warnings messages indicating that a degraded interpolation or extrapolation is performed. If the requested time is out the allowed extrapolation range, the function will return an error message.

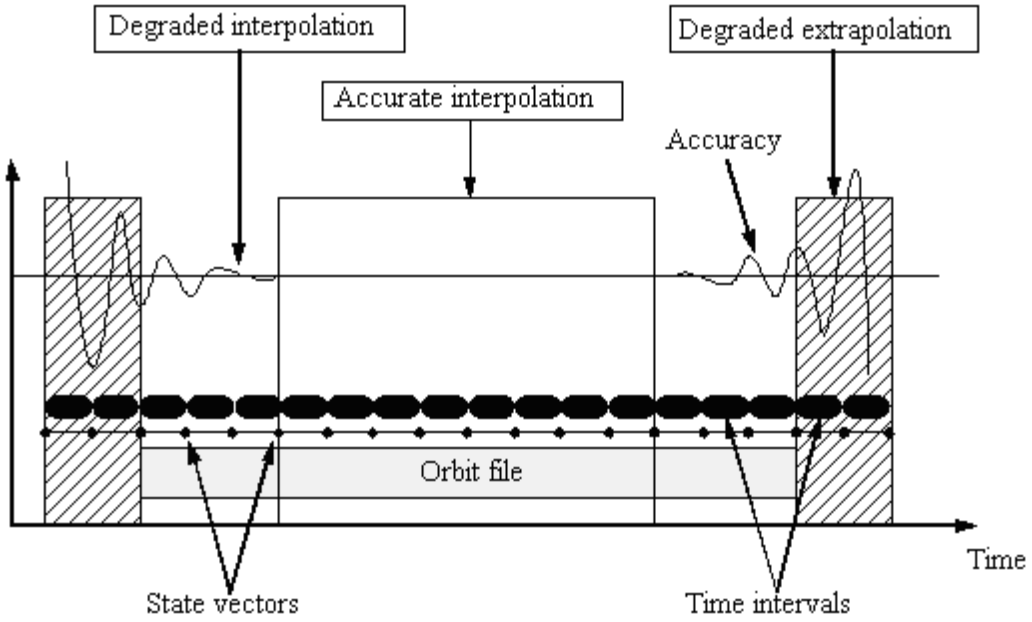


Figure 4: Performances of the interpolation algorithm

7.28.3 Calling interface

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

```
#include <explorer_orbit.h>
{
        xo_orbit_id orbit_id = {NULL};
        long mode, time_ref;
        double time, pos_out[3], vel_out[3], acc_out[3];
        long status, ierr[XO_NUM_ERR_OSV_COMPUTE];

        status = xo_osv_compute (&orbit_id, &mode, &time_ref, &time,
                                 pos_out, vel_out, acc_out, ierr);

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

        status = xo_osv_compute_run (&run_id, &mode, &time_ref, &time,
                                     pos_out, vel_out, acc_out, ierr);
}
```

7.28.4 Input parameters

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

Table 81: Input parameters of xo_osv_compute function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
mode	long *	-	Propagation model. Dummy input for current version.	-	-
time_ref	long*	-	Time reference ID	-	Complete
time	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]

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

- Time reference ID: time_ref. See [GEN_SUM].

7.28.5 Output parameters

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

Table 82: Output parameters of xo_osv_compute function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_osv_compute	long	-	Main status flag	-	-1, 0, +1
pos_out[3]	double	all	Osculating position vector at predicted time (Earth fixed CS)	m	-
vel_out[3]	double	all	Osculating velocity vector at predicted time (Earth fixed CS)	m/s	-
acc_out[3]	double	all	Osculating acceleration vector at predicted time (Earth fixed CS)	m/s^2	-
ierr[XO_NUM_ERR_PR OPAG]	long	all	Status vector	-	-

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

- Time reference ID: time_ref. See [GEN_SUM].

7.28.6 Warnings and errors

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

Table 83: Error messages of xo_osv_compute function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Could not initialise the propagation	No calculation performed	XO_CFI_OSV_COMPUTE_PROPAG_INIT_ERR	0
ERR	The internal data were not initialized	No calculation performed	XO_CFI_OSV_COMPUTE_NOT_INTERNAL_DATA_ERR R	1
ERR	Could not propagate the state vector	No calculation performed	XO_CFI_OSV_COMPUTE_PROPAG_ERR	2
ERR	Could not interpolate the state vector	No calculation performed	XO_CFI_OSV_COMPUTE_INTERPOL_ERR	3
ERR	Warnings during the propagation	No calculation performed	XO_CFI_OSV_COMPUTE_PROPAG_WARN	4
ERR	Warnings during the interpolation	No calculation performed	XO_CFI_OSV_COMPUTE_INTERPOL_WARN	5

7.28.7 Runtime performances

The following runtime performances have been measured:

Table 84: Runtime performances of xo_osv_compute function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TBD	TBD	TBD	TBD

7.29 xo_osv_compute_extra

7.29.1 Overview

This software returns ancillary results derived from an orbit state vector obtained from the **xo_osv_compute** routine (stored within the *orbit Id*). This state vector depends on which is the last function called:

- when calling **xo_osv_compute_extra** after initialising the *orbit Id*, the selected state vector is:
  - the one that is selected as seed for the propagation.
  - the first OSV stored in the orbit_id if it is initialised for interpolation.
- when calling after **xo_osv_compute**, the Cartesian orbit state vector is the one predicted at the requested time in that routine.

A description of the ancillary results may be found in the section 7.29.5.

A complete calling sequence of the computation procedure is presented in section 4.2.

7.29.2 Calling interface

The calling interface of the **xo_osv_compute_extra** CFI function is the following:

```
#include <explorer_orbit.h>
{
       xo_orbit_id orbit_id = {NULL};
       long extra_choice;
       double model_out[XO_ORBIT_EXTRA_NUM_DEP_ELEMENTS],
              extra_out[XO_ORBIT_EXTRA_NUM_INDEP_ELEMENTS];
       long status, ierr[XO_NUM_ERR_OSV_COMPUTE_EXTRA];

       status = xo_osv_compute_extra (&orbit_id, &extra_choice,
                                      model_out, extra_out, ierr);

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

       status = xo_osv_compute_run (&run_id, &extra_choice,
                                    model_out, extra_out, ierr);
}
```

7.29.3 Input parameters

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

Table 85: Input parameters of xo_osv_compute_extra

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit initialisation data	-	-
extra_choice	long *	-	Flag to allow an ancillary results choice	-	[0, 4095]

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

- Flag to select ancillary results: extra_choice. See tables below:

Table 86: Enumeration values of extra_choice input flag

Model independant	Description	Long
XO_ORBIT_EXTRA_NO_RESULTS	No extra results	0
XO_ORBIT_EXTRA_GEOLOCATION	Geolocation results	1
XO_ORBIT_EXTRA_GEOLOCATION_D	Geolocation rate results	2
XO_ORBIT_EXTRA_GEOLOCATION_2D	Geolocation rate-rate results	4
XO_ORBIT_EXTRA_GEOLOCATION_EXTRA	Geolocation extra results	8
XO_ORBIT_EXTRA_EARTH_FIXED_D	Earth fixed velocity results	16
XO_ORBIT_EXTRA_EARTH_FIXED_2D	Earth fixed acceleration results	32
XO_ORBIT_EXTRA_SUN	Sun results	64
XO_ORBIT_EXTRA_MOON	Moon results	128
XO_ORBIT_EXTRA_OSCULATING_KEPLER	Osculating keplerian elements	256
XO_ORBIT_EXTRA_INERTIAL_AUX	Inertial auxiliary results	512
Model dependant (Mean Keplerian model)	**Description**	**Long**
XO_ORBIT_EXTRA_DEP_ANX_TIMING	ANX timing results	1024
XO_ORBIT_EXTRA_DEP_MEAN_KEPLER	Mean keplerian elements	2048

To calculate all results there is an extra enumeration value, defined as the addition of all the enumeration result values:

Enumeration value	Description	Long
XO_ORBIT_EXTRA_ALL_RESULTS	All results	4095

The elements calculated in each case are shown in section 7.29.5. It is possible to select the calculation of different sets of output parameters, or to make any combination of them by adding the results enumeration desired. In order to calculate some elements it might be necessary to calculate elements which have not been explicitly requested. The function identifies internally all the dependencies and those elements are also returned in the result vectors.

7.29.4 Output parameters

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

Table 87: Output parameters of xo_osv_compute_extra

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_osv_compute_extra	long	-	Main status flag	-	-1, 0, +1
model_out[XO_ORBIT_EXTRA_NUM_DEP_ELEMENTS]	double	all	Vector of model-dependent parameters	-	-
extra_out[XO_ORBIT_EXTRA_NUM_INDEP_ELEMENTS]	double	all	Vector of model-independent parameters. It depends upon extra-choice	-	-
ierr[XO_NUM_ERR_OSV_COMPUTE_EXTRA]	long	all	Status vector	-	-

7.29.5 Results vectors

The following tables describe the set of parameters are computed with **xo_osv_compute_extra**.

The model-dependent parameters vector (note that there is an enumeration associated to the elements of the results vectors) is in Table 88. These parameters depends on the way in which the input orbit_id was initialised (the orbit mode). So, in table a column "OSV compute model" indicates the models for which that parameter can be computed. To simplify, the models are indicated with the following correspondence:

Orbit initialisation mode	OSV compute model
XO_ORBIT_INIT_ORBIT_CHANGE_MODE XO_ORBIT_INIT_OSF_MODE XO_ORBIT_INIT_OEF_OSF_MODE XO_ORBIT_INIT_POF_MODE XO_ORBIT_INIT_OEF_POF_MODE XO_ORBIT_INIT_STATE_VECTOR_MODE XO_ORBIT_INIT_POF_N_DORIS_MODE	Mean Kepler
XO_ORBIT_INIT_TLE_MODE	TLE
XO_ORBIT_INIT_POF_PRECISE_MODE XO_ORBIT_INIT_OEF_POF_PRECISE_MODE XO_ORBIT_INIT_ROF_PRECISE_MODE XO_ORBIT_INIT_DORIS_PRECISE_MODE XO_ORBIT_INIT_STATE_VECTOR_PRECISE_MODE XO_ORBIT_INIT_POF_N_DORIS_PRECISE_MODE	Precise
XO_ORBIT_INIT_ROF_MODE XO_ORBIT_INIT_DORIS_MODE	Interpolation

Table 88: Ancillary results vector. Model-dependent parameters

Result parameter	Set	Description (Reference)	Unit (Format)	Allowed Range	OSV compute model
[0] XO_ORBIT_EXTRA_DEP_NODAL_PERIOD	ANX Timing	Nodal period	s	>= 0	• Mean Kepler • Interpolation
[1] XO_ORBIT_EXTRA_DEP_UTC_CURRENT_ANX		Time of current ANX	decimal days (Processing format)	-	
[2] XO_ORBIT_EXTRA_DEP_ORBIT_NUMBER	Position in orbit[3]	Absolute Orbit Number		> 0	• Mean Kepler • TLE • Interpolation

3 These parameters are calculated only when initialising with xo_orbit_init_file and xo_orbit_init_def

[3] XO_ORBIT_EXTRA_DEP_SEC_SINCE_ANX		Time since ANX	s	>= 0 < Nodal Period	• Precise
[4:9] XO_ORBIT_EXTRA_DEP_MEAN_KEPL_A XO_ORBIT_EXTRA_DEP_MEAN_KEPL_E XO_ORBIT_EXTRA_DEP_MEAN_KEPL_I XO_ORBIT_EXTRA_DEP_MEAN_KEPL_RA XO_ORBIT_EXTRA_DEP_MEAN_KEPL_W XO_ORBIT_EXTRA_DEP_MEAN_KEPL_M	Mean Kepler	Mean Kepler elements of the propagated OSV (True of Date)	-	-	• Mean Kepler • Precise

The model-independent parameters vector (note that there is an enumeration associated to the elements of the results vectors) is in Table 89:

Table 89: Ancillary results vector. Model-independent parameters

Result parameter (res element)	Set	Description (Reference)	Unit (Format)	Allowed Range
[0] XO_ORBIT_EXTRA_GEOC_LONG	Geolocation	Geocentric longitude of satellite and SSP (EF frame)	deg	>= 0 < 360
[1] XO_ORBIT_EXTRA_GEOD_LAT		Geodetic latitude of satellite and SSP (EF frame)	deg	>= -90 <= +90
[2] XO_ORBIT_EXTRA_GEOD_ALT		Geodetic altitude of the satellite (EF frame)	m	-
[3] XO_ORBIT_EXTRA_GEOC_LONG_D	Geolocation rate	Geocentric longitude rate of satellite and SSP (EF frame)	deg/s	-
[4] XO_ORBIT_EXTRA_GEOD_LAT_D		Geodetic latitude rate of satellite and SSP (EF frame)	deg/s	-
[5] XO_ORBIT_EXTRA_GEOD_ALT_D		Geodetic altitude rate of the satellite (EF frame)	m/s	-
[6] XO_ORBIT_EXTRA_GEOC_LONG_2D	Geolocation rate rate	Geocentric longitude rate-rate of satellite and SSP (EF frame)	deg/s2	-
[7] XO_ORBIT_EXTRA_GEOD_LAT_2D		Geodetic latitude rate-rate of satellite and SSP (EF frame)	deg/s2	-
[8] XO_ORBIT_EXTRA_GEOD_ALT_2D		Geodetic altitude rate-rate of the satel lite (EF frame)	m/s2	-
[9] XO_ORBIT_EXTRA_RAD_CUR_PARALLE L_MERIDIAN	Geolocation extra	Radius of curvature parallel to meridian at the SSP (EF frame)	m	>= 0

[10] XO_ORBIT_EXTRA_RAD_CUR_ORTHO_ MERIDIAN		Radius of curvature orthogonal to meridian at the SSP (EF frame)	m	>= 0
[11] XO_ORBIT_EXTRA_RAD_CUR_ALONG_G ROUNDTRACK		Radius of curvature along groundtrack at the SSP (EF frame)	m	>= 0
[12] XO_ORBIT_EXTRA_NORTH_VEL	Earth-fixed velocity	Northward component of the velocity relative to the Earth of the SSP (Topocentric frame)	m/s	-
[13] XO_ORBIT_EXTRA_EAST_VEL		Eastward component of the velocity relative to the Earth of the SSP (Topocentric frame)	m/s	-
[14] XO_ORBIT_EXTRA_MAG_VEL		Magnitude of the velocity relative to the Earth of the SSP (Topocentric frame)	m/s	>= 0
[15] XO_ORBIT_EXTRA_AZ_VEL		Azimuth of the velocity relative to the Earth of the SSP (Topocentric frame)	deg	>= 0 < 360
[16] XO_ORBIT_EXTRA_NORTH_ACC	Earth-fixed acceleration	Northward component of the accelera tion relative to the Earth of the SSP (Topocentric frame)	m/s2	-
[17] XO_ORBIT_EXTRA_EAST_ACC		Eastward component of the accelera tion relative to the Earth of the SSP (Topocentric frame)	m/s2	-
[18] XO_ORBIT_EXTRA_GROUNDTRACK_TA NG_ACC		Groundtrack tangential component of the acceleration relative to the Earth of the SSP (Topocentric frame)	m/s2	-
[19] XO_ORBIT_EXTRA_AZ_ACC		Azimuth of the acceleration relative to the Earth of the SSP (Topocentric frame)	deg	>= 0 < 360
[20] XO_ORBIT_EXTRA_SAT_ECLIPSE_FLAG	Sun	Satellite eclipse flag 0 = No 1 = Yes		0, 1
[21] XO_ORBIT_EXTRA_SZA		Sun Zenith Angle	deg	>= 0 < 180
[22] XO_ORBIT_EXTRA_MLST		Mean local solar time at the SSP	decimal hour	>= 0 < 24
[23] XO_ORBIT_EXTRA_TLST		True local solar time at the SSP	decimal hour	>= 0 < 24
[24] XO_ORBIT_EXTRA_TRUE_SUN_RA		True Sun's (centre) right ascension (TOD frame)	deg	>= 0 < 360

[25] XO_ORBIT_EXTRA_TRUE_SUN_DEC		True Sun's (centre) declination (TOD frame)	deg	>= -90 <= +90
[26] XO_ORBIT_EXTRA_TRUE_SUN_SEMI_DI AM		True Sun's semi-diameter	deg	>= 0
[27] XO_ORBIT_EXTRA_MOON_RA	Moon	Moon's (centre) right ascension (TOD frame)	deg	>= 0 < 360
[28] XO_ORBIT_EXTRA_MOON_DEC		Moon's (centre) declination (TOD frame)	deg	>= -90 <= +90
[29] XO_ORBIT_EXTRA_MOON_SEMI_DI AM		Moon's semi-diameter	deg	>= 0
[30] XO_ORBIT_EXTRA_MOON_AREA_LI T		Area of Moon lit by Sun		>= 0 <= 1
[31:36] XO_ORBIT_EXTRA_OSC_KEPL_A XO_ORBIT_EXTRA_OSC_KEPL_E XO_ORBIT_EXTRA_OSC_KEPL_I XO_ORBIT_EXTRA_OSC_KEPL_RA XO_ORBIT_EXTRA_OSC_KEPL_W XO_ORBIT_EXTRA_OSC_KEPL_M	Osculating Kepler	Osculating Keplerian elements of the OSV (TOD frame)		
[37] XO_ORBIT_EXTRA_ORBIT_RAD	Inertial Aux	Orbit radius (TOD frame)	m	>= 0
[38] XO_ORBIT_EXTRA_RADIAL_ORB_V EL		Radial orbit velocity component (TOD frame)	m/s	-
[39] XO_ORBIT_EXTRA_TRANS_ORB_VE L		Transversal orbit velocity component (TOD frame)	m/s	-
[40] XO_ORBIT_EXTRA_ORB_VEL_MAG		Orbit velocity magnitude (TOD frame)	m/s	>= 0
[41] XO_ORBIT_EXTRA_RA_SAT		Right ascension of the satellite (TOD frame)	deg	>= 0 < 360
[42] XO_ORBIT_EXTRA_DEC_SAT		Declination of the satellite (TOD frame)	deg	>= -90 <= +90
[43] XO_ORBIT_EXTRA_EARTH_ROTATI		Earth rotation angle [H]	deg	>= 0 < 360

ON_ ANGLE				
[44] XO_ORBIT_EXTRA_RA_SAT_D		Right ascension rate of the satellite (TOD frame)	deg/s	-
[45] XO_ORBIT_EXTRA_RA_SAT_2D		Right ascension rate-rate of the satellite (TOD frame)	deg/s2	-
[46] XO_ORBIT_EXTRA_OSC_TRUE_LAT		Satellite osculating true latitude (TOD frame)	deg	>= 0 < 360
[47] XO_ORBIT_EXTRA_OSC_TRUE_LAT_D		Satellite osculating true latitude rate (TOD frame)	deg/s	-
[48] XO_ORBIT_EXTRA_OSC_TRUE_LAT_2D		Satellite osculating true latitude rate-rate (TOD frame)	deg/s2	-

7.29.6 Warnings and errors

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

Table 90: Error messages of xo_osv_compute_extra function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Could not initialise the propagation	No calculation performed	XO_CFI_OSV_COMPUTE_EXTRA_PROPAG_INIT_ERR	0
ERR	The internal data were not initialized	No calculation performed	XO_CFI_OSV_COMPUTE_EXTRA_NOT_INTERNAL_D ATA_ERR	1
ERR	Could not propagate the state vector	No calculation performed	XO_CFI_OSV_COMPUTE_EXTRA_PROPAG_ERR	2
ERR	Could not interpolate the state vector	No calculation performed	XO_CFI_OSV_COMPUTE_EXTRA_INTERPOL_ERR	3

7.29.7 Runtime performances

The following runtime performances have been measured:

Table 91: Runtime performances of xo_osv_compute_extra function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TBD	TBD	TBD	TBD

7.30 xo_orbit_to_time

7.30.1 Overview

The **xo_orbit_to_time** function converts an orbit-relative time into processing time.

7.30.2 Calling sequence of xo_orbit_to_time:

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

```
#include <explorer_orbit.h>
{
       xo_orbit_id   orbit_id = {NULL};
       long time_ref;
       long orbit, second, microsec;
       long status, ierr[XO_NUM_ERR_ORBIT_TO_TIME];
       double    time;

       status = xo_orbit_to_time (&orbit_id ,
                                  &orbit, &second, &microsec,
                                      &time_ref,
                                  &time,     ierr);

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

       status = xo_orbit_to_time_run (&run_id   ,
                                      &orbit, &second, &microsec,
                                      &time_ref,
                                      &time,     ierr);
}
```

7.30.3 Input parameters

Table 92: Input parameters for xo_orbit_to_time

C name	C type	Array Element	Description (Reference)	Unit (For mat)	Allowed Range
orbit_id	xo_orbit _id*	-	Structure that contains the orbit data	-	-
orbit	long*		Absolute orbit number		> 0
second	long*		Seconds since ascending node	s	>= 0 <orbital period
microsec	long*		Micro seconds within second	µs	0 =< =< 999999
time_ref	long*		Time reference ID	-	Complete

7.30.4 Output parameters

Table 93: Output parameters for xo_orbit_to_time

C name	C type	Array Eleme nt	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_to_time	long		Main status flag		-1, 0, 1
time	double*		Resulting time	Dedimal days (processing format)	[-18262.0, +36519.0]
ierr[XO_NUM_ERR _ORBIT_TO_TIME]	long		Error status flags		

7.30.5 Warnings and errors

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

Table 94: Error messages of xo_orbit_to_time function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	Computation not performed	XO_CFI_ORBIT_TO_TIME_FLAG_ERR	0
ERR	Input incorrect: negative orbit number	Computation not performed	XO_CFI_ORBIT_TO_TIME_ORB_NUM_1ST_ERR	1
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_TO_TIME_ORBIT_STATUS_ERR	2
ERR	Seconds and microseconds greater than nodal period	Computation not performed	XO_CFI_ORBIT_TO_TIME_SEC_MICROSEC_ERR	3
ERR	Requested orbit less than the first orbital change	Computation not performed	XO_CFI_ORBIT_TO_TIME_ORB_ERR	4
ERR	Input incorrect: negative number of seconds	Computation not performed	XO_CFI_ORBIT_TO_TIME_SEC_ERR	5
ERR	Input incorrect: number of microseconds out of range	Computation not performed	XO_CFI_ORBIT_TO_TIME_MICROSEC_ERR	6
ERR	Error computing time.	Computation not performed	XO_CFI_ORBIT_TO_TIME_COMPUTE_ERR	7
ERR	Could not make a time transformation	Computation not performed	XO_CFI_ORBIT_TO_TIME_TIME_CHANGE_ERR	8

7.30.6 Runtime performances

The following runtime performances have been measured:

Table 95: Runtime performances of xo_orbit_to_time function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.073	0.012	0.016	0.0022

7.30.7 Executable Program

The conversion from orbit to time described before can be carried out by the **orbit_to_time** executable program as follows:

orbit_to_time **-sat** satellite_name
-file Orbit file
-tref time_ref
-orb orbit
-anx anx_time (seconds)

[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
{ (**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) }) **}**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT 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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *time_ref* and *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- Data for initialising the time references are needed only when using an Orbit Scenario file. For other files the data is optional. In that case, if the iniialization parameters are not provided, the time correlations are initialised with the input orbit file.

  The inputs needed for time initialiazation are provided in the last three lines of parameters. Note that only one set of parameters should be introduced:

  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)

Example:

`orbit_to_time` **-sat** CRYOSAT **-file** EARTH_EXPLORER_FPO **-tref** UTC
-orb 1001 **-anx** 0.0 **-`show` -v**

7.31 xo_time_to_orbit

7.31.1 Overview

The **xo_time_to_orbit** function converts an orbit-relative time into processing time.

7.31.2 Calling sequence of xo_time_to_orbit

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

```
#include <explorer_orbit.h>
{
       xo_orbit_id   orbit_id = {NULL};
       long  time_ref;
       long orbit, second, microsec;
       long status, ierr[XO_NUM_ERR_ORBIT_TO_TIME];
       double    time;

       status = xo_time_to_orbit ( &orbit_id    ,
                                   &time_ref, &time,
                                   &orbit, &second, &microsec,
                                   ierr);

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

       status = xo_time_to_orbit_run ( &run_id  ,
                                       &time_ref, &time,
                                       &orbit, &second, &microsec,
                                       ierr);
}
```

7.31.3 Input parameters

Table 96: Input parameters for xo_time_to_orbit function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
time_ref	long*		Time reference ID	-	Complete
time	double*		Requested time	Decimal days (processing format)	[-18262.0, +36519.0]

7.31.4 Output parameters

Table 97: Output parameters for xo_time_to_orbit

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_time_to_orbit	long		Main status flag		-1, 0, 1
orbit	long*		Absolute orbit number		> 0
second	long*		Seconds since ascending node	s	>= 0 <orbital period
microsec	long*		Micro seconds within second	µs	0 =< =< 999999
ierr[XO_NUM_ERR_TIME_TO_ORBIT]	long		Error status flags		

7.31.5 Warnings and errors

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

Table 98: Error messages of xo_time_to_orbit function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Wrong input flag	Computation not performed	XO_CFI_TIME_TO_ORBIT_FLAG_ERR	0
ERR	Orbit Id. was not initialized.	Computation not performed	XO_CFI_TIME_TO_ORBIT_ORBIT_STATUS_ERR	1
ERR	Input incorrect: time out of range.	Computation not performed	XO_CFI_TIME_TO_ORBIT_TIME_ERR	2
ERR	Input time smaller than the first ANX time.	Computation not performed	XO_CFI_TIME_TO_ORBIT_BEFORE_RANGE_ERR	3
ERR	Could not compute the orbit number.	Computation not performed	XO_CFI_TIME_TO_ORBIT_COMPUTE_ERR	4
ERR	The current orbit initialization does not allow to compute the time.	Computation not performed	XO_CFI_TIME_TO_ORBIT_WRONG_ORBIT_MODE_ERR	5
WARN	Input time before first orbit.	Computation performed	XO_CFI_TIME_TO_ORBIT_TIME_BEFORE_RANGE_WARN	6
WARN	Input time after first orbit.	Computation performed	XO_CFI_TIME_TO_ORBIT_TIME_AFTER_RANGE_WARN	7
WARN	Orbit number computed with warnings.	Computation performed	XO_CFI_TIME_TO_ORBIT_COMPUTE_WARN	8

7.31.6 Runtime performances

The following runtime performances have been measured:

Table 99: Runtime performances of xo_time_to_orbit function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.397	0.171	0.288	0.058

7.31.7 Executable Program

The conversion from time to orbit described before can be carried out by the **time_to_orbit** executable program as follows:

time_to_orbit **-sat** satellite_name
-file Orbit file
-tref time_ref
{**-time** time (days) | **-atime** time (CCSDSA format)}
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
{ (**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) }) **}**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT 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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *time_ref* and *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- Data for initialising the time references are needed only when using an Orbit Scenario file. For other files the data are optional. In that case, if the iniialization parameters are not provided, the time correlations are initialised with the input orbit file

  The inputs needed for time initialization are provided in the last three lines of parameters. Note that only one set of parameters should be introduced:

  – TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)

- A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)

Example:

time_to_orbit **-sat** CRYOSAT **-file** EARTH_EXPLORER_FPO **-tref** UTC **-time** -2010.108657407 **-show -v**

7.32 xo_orbit_info

7.32.1 Overview

The **xo_orbit_info** function retrieves from the orbit initialisation, information related with a certain orbit (specified by means of absolute orbit number).

7.32.2 Calling sequence of xo_orbit_info

For C programs, the call to **xo_orbit_info** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
       xo_orbit_id  orbit_id = {NULL};
       long         abs_orbit;
       long         ierr[XO_NUM_ERR_ORBIT_INFO], status;
       double       result_vector[XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS];

       status = xo_orbit_info (&orbit_id,
                               &abs_orbit,
                               result_vector, ierr);

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

       status = xo_orbit_info_run (&run_id,
                                   &abs_orbit,
                                   result_vector, ierr);
}
```

7.32.3 Input parameters

Table 100: Input parameters for xo_orbit_info

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data.	-	-
abs_orbit	long *		Absolute orbit number		within orbit_id range

7.32.4 Output parameters

Table 101: Output parameters for xo_orbit_info

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_info	long		Main status flag,		-1, 0, 1
result_vector[XO_ORBIT_INFO_EXTRA_NUM_ELEMENTS]	double	[0]	repeat_cycle[4]	days	>0
		[1]	cycle_lengtha	orbits	>0
		[2]	MLST drifta	s/day	
		[3]	MLST[5]	hours	>0 <24
		[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	
ierr[XO_ORBIT_INFO_FROM_ABS]	long	all	Error status flags		

4 This parameter is only computed if the input orbit_id was computed either with an Orbit Sceanrio file using xo_orbit_init_file or with xo_orbit_init_def

5 This parameter is not computed if the input orbit_id was computed using a Restituted Orbit file or a DORIS file

7.32.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_info** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_ORBIT software library **xo_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 pv_utcanx CFI function by calling the function of the EO_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 102: Error messages of xo_orbit_info function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. No initialised.	Computation not performed	XO_CFI_ORBIT_INFO _ORBIT_INIT_ERR	0
ERR	Orbit out of initialised limits.	Computation not performed	XO_CFI_ORBIT_INFO _OUT_OF_LIMITS_ER R	1
ERR	Could not compute extra results	Computation not performed	XO_CFI_ORBIT_INFO _RESULTS_ERR	2

7.32.6 Runtime performances

The following runtime performances have been measured:

Table 103: Runtime performances of xo_orbit_info function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
1.362	0.337	0.4200	0.155

7.33 xo_orbit_rel_from_abs

7.33.1 Overview

The **xo_orbit_rel_from_abs** function retrieves from an Orbit Scenario File (previously initialised through the *orbit Id*) the relative orbit corresponding to a given absolute orbit number.

7.33.2 Calling sequence of xo_orbit_rel_from_abs

For C programs, the call to **xo_orbit_rel_from_abs** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
       xo_orbit_id    orbit_id = {NULL};
       long                abs_orbit, rel_orbit, cycle, phase;
       long            ierr[XO_NUM_ERR_ORBIT_REL_FROM_ABS], status;

       status = xo_orbit_rel_from_abs (&orbit_id,
                                       &abs_orbit,
                                       &rel_orbit, &cycle,
                                       &phase, ierr);

       /* Or, using the run_id */
       long run_id;
       status = xo_orbit_rel_from_abs_run (&run_id,
                                           &abs_orbit,
                                           &rel_orbit, &cycle,
                                           &phase, ierr);
}
```

7.33.3 Input parameters

Table 104: Input parameters for xo_orbit_rel_from_abs

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
abs_orbit	long *		Absolute orbit number		within orbit_id range

7.33.4 Output parameters

Table 105: Output parameters for xo_orbit_rel_from_abs

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_rel_from_abs	long		Main status flag,		-1, 0, 1
rel_orbit	long *		Relative orbit number		
cycle	long *		Cycle number		
phase	long *		Phase number		
ierr[XO_ORBIT_REL_FROM_ABS]	long	all	Error status flags		

7.33.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_rel_from_abs** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_ORBIT software library **xo_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 pv_utcanx CFI function by calling the function of the EO_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 106: Error messages of xo_orbit_rel_from_abs function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_REL_FROM_ABS_ORBIT_INIT_ERR	0
ERR	The relative orbit could not be computed with the current orbit initialization.	Computation not performed	XO_CFI_ORBIT_REL_FROM_ABS_ORBIT_WRONG_MODE_ERR	1
ERR	Wrong input orbit number	Computation not performed	XO_CFI_ORBIT_REL_FROM_ABS_WRONG_ORBIT	2

7.33.6 Runtime performances

The following runtime performances have been measured:

Table 107: Runtime performances of xo_orbit_rel_from_abs function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.0094	0.0018	0.002	0.0002

7.34 xo_orbit_abs_from_rel

7.34.1 Overview

The **xo_orbit_abs_from_rel** function retrieves from an Orbit Scenario File (previously initialised through the *orbit Id*) the absolute orbit corresponding to a given relative orbit number and cycle.

7.34.2 Calling sequence of xo_orbit_abs_from_rel

For C programs, the call to **xo_orbit_abs_from_rel** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
       xo_orbit_id  orbit_id = {NULL};
       long         abs_orbit, rel_orbit, cycle, phase;
       long         ierr[XO_NUM_ERR_ORBIT_ABS_FROM_REL], status;

       status = xo_orbit_abs_from_rel (&orbit_id,
                                       &rel_orbit, &cycle,
                                       &abs_orbit, &phase, ierr);

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

       status = xo_orbit_abs_from_rel_run (&run_id,
                                           &rel_orbit, &cycle,
                                           &abs_orbit, &phase, ierr);
}
```

7.34.3 Input parameters

Table 108: Input parameters for xo_orbit_abs_from_rel

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
rel_orbit	long *		Relative orbit number		
cycle	long *		Cycle number		

7.34.4 Output parameters

Table 109: Output parameters for xo_orbit_abs_from_rel

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_abs_from_rel	long		Main status flag,		-1, 0, 1
abs_orbit	long *		Absolute orbit number		within orbit_id range
phase	long *		Phase number		
ierr[XO_ORBIT_ABS_FROM_REL]	long	all	Error status flags		

7.34.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_abs_from_rel** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_ORBIT software library **xo_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 pv_utcanx CFI function by calling the function of the EO_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 110: Error messages of xo_orbit_abs_from_rel function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_REL_ORBIT_INIT_ERR	0
ERR	The orbit numbers could not be computed with the current orbit initialization.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_REL_ORBIT_WRONG_MODE_ERR	1
ERR	Wrong input relative orbit and/or cycle.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_REL_INPUT_PARAMETER_ERR	2

7.34.6 Runtime performances

The following runtime performances have been measured:

Table 111: Runtime performances of xo_orbit_abs_from_rel function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.0098	0.0016	0.0022	0.0002

7.35 xo_orbit_abs_from_phase

7.35.1 Overview

The **xo_orbit_abs_from_phase** function retrieves from an Orbit Scenario File (previously initialised through the *orbit Id*) the absolute orbit corresponding to a given phase.

7.35.2 Calling sequence of xo_orbit_abs_from_phase

For C programs, the call to **xo_orbit_abs_from_phase** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
       xo_orbit_id  orbit_id = {NULL};
       long         abs_orbit, rel_orbit, cycle, phase;
       long         ierr[XO_NUM_ERR_ORBIT_ABS_FROM_REL], status;

       status = xo_orbit_abs_from_phase (&orbit_id,
                                         &phase,
                                         &abs_orbit,
                                         &rel_orbit, &cycle,
                                         ierr);

       /* Or, using the run_id */
       long run_id;
       status = xo_orbit_abs_from_phase_run (&run_id,
                                             &phase,
                                             &abs_orbit,
                                             &rel_orbit, &cycle,
                                             ierr);
}
```

7.35.3 Input parameters

Table 112: Input parameters for xo_orbit_abs_from_phase

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
phase	long *		Phase number		

7.35.4 Output parameters

Table 113: Output parameters for xo_orbit_abs_from_phase

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_orbit_abs_from_phase	long		Main status flag,		-1, 0, 1
abs_orbit	long *		Absolute orbit number		within orbit_id range
rel_orbit	long *		Relative orbit number		
cycle	long *		Cycle number		
ierr[XO_ORBIT_ABS_FROM_PHASE]	long	all	Error status flags		

7.35.5 Warnings and errors

Next table lists the possible error messages that can be returned by the **xo_orbit_abs_from_phase** CFI function after translating the returned status vector into the equivalent list of error messages by calling the function of the EO_ORBIT software library **xo_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 pv_utcanx CFI function by calling the function of the EO_ORBIT software library **xo_get_code** (see [GEN_SUM]).

Table 114: Error messages of xo_orbit_abs_from_phase function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Orbit Id. is not initialised.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_PHASE_ORBIT_INIT_ERR	0
ERR	The orbit numbers could not be computed with the current orbit initialization.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_PHASE_ORBIT_WRONG_MODE_ERR	1
ERR	Wrong input phase number.	Computation not performed	XO_CFI_ORBIT_ABS_FROM_PHASE_INPUT_PARAMETER_ERR	2

7.35.6 Runtime performances

The following runtime performances have been measured:

Table 115: Runtime performances of xo_orbit_abs_from_phase function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
0.01	0.0016	0.002	0.0004

7.36 xo_osv_to_tle

7.36.1 Overview

The **xo_osv_to_tle** function generates a TLE by fitting the set of orbit state vectors stored in the orbit_id. This set of OSVs are selected from the input orbit_id for the orbit/time requested range. Note that it is possible to convert only one OSV if:

- the requeted time range only contains an OSV.
- the start orbit equal to the stop orbit.

Note: Currently, only one OSV can be converted to TLE. In case of introducing a time/orbit range, the first OSV in that range is selected for the conversion.

7.36.2 Calling sequence of xo_osv_to_tle

For C programs, the call to **xo_osv_to_tle** is (input parameters are underlined, some may be input or output depending on the calling mode):

```
#include <explorer_orbit.h>
{
       xo_orbit_id    orbit_id = {NULL};
       xd_tle_rec     tle_rec;
       long                time_mode, time_ref, orbit0, orbit1;
      double           time0, time1;
       long             ierr[XO_NUM_ERR_OSV_TO_TLE], status;

       status = xo_osv_to_tle (&orbit_id,
                               &time_mode, &time_ref,
                               &time0, &time1,
                               &orbit0, &orbit1,
                               /* outputs */
                               &tle_rec,
                               ierr);
}
```

7.36.3 Input parameters

Table 116: Input parameters for xo_osv_to_tle

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
orbit_id	xo_orbit_id*	-	Structure that contains the orbit data	-	-
time_mode	long	-	time/orbit selection mode. For the XL_SEL_DEFAULT mode, the whole range of orbits stored in the orbit_id is selected	-	XO_SEL_TIME XO_SEL_ORBIT XO_SEL_DEFAUL T
time_ref	long	-	time reference (only used if time_mode is XO_SEL_TIME	-	Complete
time0	double	-	Start time	days	Start validity time for the orbit_id
time1	double	-	Output time	days	Stop validity time for the orbit_id
orbit0	long	-	Start orbit	-	First orbit stored in the orbit_id
orbit1	long	-	Stop orbit	-	Last orbit stored in the orbit_id

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

- time_mode: See [LIB_SUM], section 6.2 (Time Initialization)
- time_ref: See [LIB_SUM], section 6.2 (Time reference).

7.36.4 Output parameters

Table 117: Output parameters for xo_osv_to_tle

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_osv_to_tle	long	-	Main status flag	-	-1, 0, 1
tle_rec	xd_tle_rec	-	TLE record data	-	-
ierr	long	all	error array	-	-

7.36.5 Warnings and errors

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

Table 118: Error messages of xo_osv_to_tle function

Error type	Error message	Cause and impact	Error Code	Error No
ERR	Input orbit_id is initialised with an incorrect model	Computation not performed	XO_CFI_OSV_TO_TLE_WRONG_FILE_MODEL_ERR	0
ERR	The input time/orbit interval is not correct	Computation not performed	XO_CFI_OSV_TO_TLE_WRONG_INPUT_INTERVAL_ERR	1
ERR	Error in a time transformation	Computation not performed	XO_CFI_OSV_TO_TLE_TIME_TRANS_ERR	2
ERR	Incorrect input time mode	Computation not performed	XO_CFI_OSV_TO_TLE_WRONG_TIME_MODEL_ERR	3
ERR	Could not change from EF CS to TEME CS	Computation not performed	XO_CFI_OSV_TO_TLE_CHANGE_CS_ERR	4
ERR	Could not get keplerian elements for absolute orbit	Computation not performed	XO_CFI_OSV_TO_TLE_CART_TO_KEPLER_ERR	5

7.36.6 Runtime performances

The following runtime performances have been measured:

Table 119: Runtime performances of xo_osv_to_tle function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TBD	TBD	TBD	TBD

7.37 xo_gen_osf_create

7.37.1 Overview

The **xo_gen_osf_create** CFI function creates a reference Orbit Scenario File (OSF) with one orbit change data structure using only user inputs in the calling interface. This data structure characterizes the reference orbit by means of the following parameters:

- Absolute orbit number
- Relative orbit number
- Cycle number
- Phase number
- Repeat cycle (days)
- Cycle length (orbits)
- Ascending crossing node longitude
- Mean local solar time of the ascending crossing node
- Mean local solar time drift (seconds per day)
- Time of the ascending crossing node (TAI, UTC and UT1)

7.37.2 Calling interface

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

```
#include <explorer_orbit.h>
{
      long sat_id;
      xl_model_id model_id = {NULL};
      xl_time_id time_id = {NULL};
      long abs_orbit_number, cycle_number, phase_number,
           repeat_cycle, cycle_length, drift_mode, version_number;
      double anx_long, inclination, mlst_drift, mlst, date;
      char output_dir[XD_MAX_STR], output_filename[XD_MAX_STR];
      char *file_class, *fh_system;
      long status, ierr[XO_ERR_VECTOR_MAX_LENGTH];
      status = xo_gen_osf_create (&sat_id, &model_id, &time_id,
                                  &abs_orbit_number,
                                  &cycle_number, &phase_number,
                                  &repeat_cycle, &cycle_length,
                                  &anx_long, &drift_mode,
```

```
                                        &inclination, &mlst_drift,
                                        &mlst, &date,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                       ierr);

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

        status = xo_gen_osf_create_run (&run_id, &abs_orbit_number,
                                        &cycle_number, &phase_number,
                                        &repeat_cycle, &cycle_length,
                                        &anx_long, &drift_mode,
                                        &inclination, &mlst_drift,
                                        &mlst, &date,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                       ierr);
    }
```

7.37.3 Input parameters

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

Table 120: Input parameters of xo_gen_osf_create function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id	-	Model ID	-	Complete
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
abs_orbit_number	long*	-	Orbit number in OSF first orbit change	-	>= 1
cycle_number	long*	-	Cycle number in OSF first orbit change	-	>= 1
phase_number	long*	-	Phase number in OSF first orbit change	-	>= 1
repeat_cycle	long*	-	Repeat cycle of the reference orbit	days	>= 1
cycle_length	long*	-	Cycle length of the reference orbit	orbits	>= 14
anx_long	double*	-	Reference orbit ascending node crossing longitude	deg	[-180, 180]
drift_mode	long*	-	Flag to select between drift in mean local solar time and inclination as input characterization of the reference orbit	-	[0,1]
inclination	double*	-	If *drift_mode* = *XO_NOSUNSYNC_INCLINATION* Inclination of the reference orbit	deg	[0,180]
mlst_drift	double*	-	If *drift_mode* = *XO_NOSUNSYNC_DRIFT* Drift in mean local solar time of the reference orbit: · MLST[N+1]=MLST[N]+MLST drift	seconds/day	TBD
mlst	double*	-	Mean local solar time at ascending node	decimal hours	[0,24)
date	double*	-	ANX date	decimal days	-
output_dir	char*	-	Directory where the resulting OSF is written (if empty (i.e. “”),	-	-

			the current directory is used)		
output_filename	char*	-	Output OSF name if empty (i.e. “”), the software will generate the filename according to file name specification pre sented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output Orbit file	-	-
version_number	long*	-	Version number of output Orbit file	-	>= 1
fh_system	char*	-	System field of the output Orbit file fixed header	-	-

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

- Satellite ID: sat_id.
- Drift mode: mlst_drift.

This CFI can generate Orbit Scenario Files for both sun-synchronous orbits and quasi-sun-synchronous orbits.

Use `drift_mode=XO_NOSUNSYNC_DRIFT` and `mlst_drift = 0.0` for a sun-synchronous orbit.

Use any other combination for the general case of quasi-sun-synchronous orbit.

7.37.4 Output parameters

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

Table 121: Output parameters of xo_gen_osf_create function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
output_filename	char*	-	Name for output file. This is only an output parameter when it is empty (i.e. “”; see description of this parameter in Table 120)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.37.5 Warnings and errors

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

Table 122: Error messages of xo_gen_osf_create function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input values	Wrong value of one or more of the following input parameters: abs_orbit_number, cycle_number, phase_number, repeat_cycle, cycle_length, mlst Computation not performed	XO_CFI_GEN_OSF_CREATE_INPUTS_ERR	0
ERR	Time ID is not initialized	Time correllations were not initialized. Computation not performed	XO_CFI_GEN_OSF_CREATE_TIME_INIT_ERR	1
ERR	Memory allocation error	Memory allocation error for the orbit change data structure Computation not performed	XO_CFI_GEN_OSF_CREATE_ALLOC_ERR	2
ERR	Wrong drift mode	Wrong drift mode flag value for charaterization of non-sun.synchronous orbits Computation not performed	XO_CFI_GEN_OSF_CREATE_DRIFT_MODE_ERR	3
ERR	Error calculating MLST drift	Error calculating MLST drift from inclination Computation not performed	XO_CFI_GEN_OSF_CREATE_DRIFT_CALC_ERR	4
ERR	Error calculating UTC of ANX	Error calculating the UTC time of the orbit ascending	XO_CFI_GEN_OSF_CREATE_UTC_CALC_ERR	5

		node Computation not performed		
ERR	Error calculating TAI of ANX	Error calculating the TAI time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_CREATE_TAI_CALC_ERR	6
ERR	Error calculating UT1 of ANX	Error calculating the UT1 time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_CREATE_UT1_CALC_ERR	7
ERR	Error calculating the Fixed Header data	Error getting the data for the Fixed Header. Computation not performed	XO_CFI_GEN_OSF_CREATE_GET_FH_ERR	8
ERR	Error writing file to disk	Error writing the data structure to a file on disk Computation not performed	XO_CFI_GEN_OSF_CREATE_WRITE_ERR	9

7.37.6 Runtime performances

The following runtime performance has been measured.

Table 123: Runtime performances of xo_gen_osf_create function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
9.8	6.9	7.6	5.6

7.37.7 Executable Program

The **gen_osf_create** executable program can be called from a Unix shell as:

gen_osf_create **-sat** satellite_name
-orbit abs_orbit_number
-cyc cycle_number
-pha phase_number
-repcyc repeat_cycle (days)
-cyclen cycle_length (orbits)
-anx anx_long (deg)
{**-mlstdr** mlst_drift| **-inc** inclination}
-mlst mlst (decimal hours)
-date anx_date
[**-dir** dir_name] (current directory by default)
[**-osf** name of the orbit scenario file] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-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.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT Verbose mode.
- [**-v**] option for Verbose mode for all libraries (default is Silent)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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- The last three lines of parameters are used to initialize the time references. In order to do this, only one set of parameters should be introduced:

- TAI, GPS, UTC and UT1 input times

- A file with time reference data, the time mode, the time reference name and the time range

Example:

```
gen_osf_create -sat CRYOSAT -orbit 1 -cyc 1 -pha 1 -repcyc 2
               -cyclen 29 -inc 92 -mlst 21 -date 790 -anx 130
               -dir ./gen_osf -osf mpl_orb_sc_at_302
               -tai -1100.1 -utc -1100.099595
               -ut1 -1100.0995914352 -gps -1100.0997801
```

7.38 xo_gen_osf_append_orbit_change

7.38.1 Overview

The **xo_gen_osf_append_orbit_change** CFI function appends an orbit change to an existing reference Orbit Scenario File (OSF). The user must provide in the calling interface the name of the existing OSF, the parameters describing the new orbit change and the output file name where the old OSF with the appended orbit change will be written. No output file is generated if the resulting orbit is discontinuous in terms of ascending node longitude, mean local solar time.

7.38.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long sat_id;
        xl_time_id time_id = {NULL};
        xl_model_id model_id = {NULL};
        long abs_orbit_number,          repeat_cycle, cycle_length,
             drift_mode, phase_increment, version_number;
        double anx_long, inclination, mlst_drift, mlst;
        char input_filename[XD_MAX_STR],
             output_dir[XD_MAX_STR], output_filename[XD_MAX_STR];
        char *file_class, *fh_system;
        long status, ierr[XO_ERR_VECTOR_MAX_LENGTH];
        status = xo_gen_osf_append_orbit_change (&sat_id, &model_id,
                                        &time_id,
                                        &input_filename,
                                        &abs_orbit_number,
                                        &repeat_cycle, &cycle_length,
                                        &anx_long, &drift_mode,
                                        &inclination, &mlst_drift,
                                        &mlst, &phase_increment,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                       ierr);

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

```
        status = xo_gen_osf_append_orbit_change_run (&run_id,
                                        &input_filename,
                                        &abs_orbit_number,
                                        &repeat_cycle, &cycle_length,
                                        &anx_long, &drift_mode,
                                        &inclination, &mlst_drift,
                                        &mlst, &phase_increment,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                        ierr);
    }
```

7.38.3 Input parameters

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

Table 124: Input parameters of xo_gen_osf_append_orbit_change function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	long *	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
input_filename	char*	-	Input OSF to which the orbit change is appended		
abs_orbit_number	long*	-	Absolute orbit number of the new orbit change	-	> abs orbit number in input OSF last orbit change
repeat_cycle	long*	-	Repeat cycle of the new reference orbit	days	>= 1
cycle_length	long*	-	Cycle length of the new reference orbit	orbits	>= 14
anx_long	double*	-	Requested orbit ascending node crossing longitude	deg	[-180, 180]
drift_mode	long*	-	Flag to select between drift in mean local solar time and inclination as input characterization of the reference orbit	-	[0,1]
inclination	double*	-	If *drift_mode* = *XO_NOSUNSYNC_INCLINATION* Inclination of the reference orbit	deg	[0,180]
mlst_drift	double*	-	If *drift_mode* = *XO_NOSUNSYNC_DRIFT* Drift in mean local solar time of the reference orbit: · MLST[N+1]=MLST[N]+MLST drift	seconds/day	TBD
mlst	double*	-	Mean local solar time at ascending node	decimal hours	[0,24)

phase_increment	long*	-	If 1 then
			phase [N+1] = phase [N] + 1
			If 0 then
				-	[0, 1]

			phase [N+1] = phase [N]		
output_dir	char*	-	Directory where the resulting OSF is written (if empty (i.e. “”), the current directory is used)	-	-
output_filename	char*	-	Output OSF name if empty (i.e. “”), the software will generate the filename according to file name specification pre sented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output Orbit file	-	-
version_number	long*	-	Version number of output Orbit file	-	>= 1
fh_system	char*	-	System field of the output Orbit file fixed header	-	-

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

- Satellite ID: sat_id. See [GEN_SUM].
- Drift mode: mlst_drift.
- Phase increment.

This CFI can append orbit changes for both sun-synchronous orbits and quasi-sun-synchronous orbits.

Use `drift_mode=XO_NOSUNSYNC_DRIFT` and `mlst_drift = 0.0` for a sun-synchronous orbit.

Use any other combination for the general case of quasi-sun-synchronous orbit.

7.38.4 Output parameters

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

Table 125: Output parameters of xo_gen_osf_append_orbit_change function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
output_filename	char*	-	Name for output file. This is only an output parameter when it is empty (i.e. “”; see description of this parameter in Table 124)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.38.5 Warnings and errors

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

Table 126: Error messages of xo_gen_osf_append_orbit_change function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input values	Wrong value of one or more of the following input parameters: abs_orbit_number, repeat_cycle, cycle_length, mlst, phase_increment Computation not performed	XO_CFI_GEN_OSF_APPEND_INPUTS_ERR	0
ERR	Time ID is not initialized	Time correlations were not initialized. Computation not performed	XO_CFI_GEN_OSF_APPEND_TIME_INIT_ERR	1
ERR	Cannot read input OSF	Computation not performed	XO_CFI_GEN_OSF_APPEND_READ_IN_OSF_ERR	2
ERR	ANX long jump larger than allowed	Requested ANX long leads to an orbit discontinuity	XO_CFI_GEN_OSF_APPEND_ANX_LONG_ERR	3
ERR	MLST jump larger than allowed	Requested MLST leads to an orbit discontinuity	XO_CFI_GEN_OSF_APPEND_MLST_ERR	4
ERR	Wrong drift mode	Wrong drift mode flag value for charaterization of non-sun.synchronous orbits Computation not performed	XO_CFI_GEN_OSF_APPEND_DRIFT_MODE_ERR	5
ERR	Error calculating MLST drift	Error calculating MLST drift from inclination Computation not performed	XO_CFI_GEN_OSF_APPEND_DRIFT_CALC_ERR	6
ERR	Error calculating UTC of ANX	Error calculating the UTC time of the orbit ascending node	XO_CFI_GEN_OSF_APPEND_UTC_CALC_ERR	7

		Computation not performed		
ERR	Error calculating TAI of ANX	Error calculating the TAI time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_APPEND_TAI_CALC_ERR	8
ERR	Error calculating UT1 of ANX	Error calculating the UT1 time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_APPEND_UT1_CALC_ERR	9
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_OSF_APPEND_ALLOC_ERR	10
ERR	Error calculating the Fixed Header data	Computation not performed	XO_CFI_GEN_OSF_APPEND_GET_FH_ERR	11
ERR	Error writing file to disk	Error writing the data structure to a file on disk Computation not performed	XO_CFI_GEN_OSF_APPEND_WRITE_ERR	12

7.38.6 Runtime performances

The following runtime performance has been measured.

Table 127: Runtime performances of xo_gen_osf_append_orbit_change function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
14.5	8.8	11.2	6

7.38.7 Executable Program

The **gen_osf_append_orbit_change** executable program can be called from a Unix shell as:

gen_osf_append_orbit_change **-sat** satellite_name
-inosf input_filename
-orbit abs_orbit_number
-repcyc repeat_cycle(days)
-cyclen cycle_length(orbits)
-anx anx_long(deg)
{ **-mlstdr** mlst_drift | **-inc** inclination }
-mlst mlst
[**-phinc**]
[**-dir** output_dir] (current directory by default)
[**-osf** output_filename] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
{ (**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) }) **}**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-phinc**] option for phase_increment. Default value for phase_increment is xo_NO_PHASE_INCREMENT. When the option is written, phase_increment is xo_PHASE_INCREMENT.

- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT 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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- The last three lines of parameters are used to initialize the time references. In order to do this, only one set of parameters should be introduced:
  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)

Example:

```
gen_osf_append_orbit_change -sat CRYOSAT
     -inosf CS_TEST_MPL_ORBREF_20020301T122001_99999999T999999_0001.EEF
     -orbit 30 -repcyc 366 -cyclen 5344 -anx 129.9986 -mlst 20.90083
     -inc 92 -dir ./gen_osf -osf mpl_orb_sc_at_303
     -tai -1100.1 -utc -1100.099595
     -ut1 -1100.0995914352 -gps -1100.0997801
```

7.39 xo_gen_osf_change_repeat_cycle

7.39.1 Overview

Given a reference orbit from an existing OSF and a new target orbit (repeat cycle, cycle length, ascending node longitude and inclination or mean local solar time drift), the **xo_gen_osf_change_repeat_cycle** CFI function finds an optimum orbit change such that the target orbit can be reached from the found orbit change. This function will write a new OSF with the found orbit change appended to the content of the old OSF.

7.39.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long sat_id;
        xl_model_id model_id = {NULL};
        xl_time_id time_id = {NULL};
        long abs_orbit_number, search_direction, repeat_cycle,
             cycle_length, drift_mode, phase_increment, version_number;
        double anx_long, inclination, mlst_drift;
        char input_filename[XD_MAX_STR],
             output_dir[XD_MAX_STR], output_filename[XD_MAX_STR];
        char *file_class, *fh_system;
        long status, ierr[XO_ERR_VECTOR_MAX_LENGTH];

        status = xo_gen_osf_change_repeat_cycle (&sat_id, &model_id,
                                        &time_id, &input_filename,
                                        &abs_orbit_number,
                                        &search_direction,
                                        &repeat_cycle, &cycle_length,
                                        &anx_long, &drift_mode,
                                        &inclination, &mlst_drift,
                                        &phase_increment,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                        ierr);

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

```
        long run_id;

        status = xo_gen_osf_change_repeat_cycle_run (&run_id,
                                    &input_filename, &abs_orbit_number,
                                    &search_direction,
                                    &repeat_cycle, &cycle_length,
                                    &anx_long, &drift_mode,
                                    &inclination, &mlst_drift,
                                    &phase_increment,
                                    output_dir, output_filename,
                                    file_class, &version_number,
                                    fh_system,
                                   ierr);
    }
```

7.39.3 Input parameters

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

Table 128: Input parameters of xo_gen_osf_change_repeat_cycle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
input_filename	char*	-	Input OSF to which the orbit change is appended		
abs_orbit_number	long*	-	Absolute orbit number from which the optimum transition search starts	-	> abs orbit number in input OSF last orbit change
search_direction	long*	-	Search for optimum transition after or before abs_orbit_number		{-1, 1}
repeat_cycle	long*	-	Repeat cycle of the new reference orbit	days	>= 1
cycle_length	long*	-	Cycle length of the new reference orbit	orbits	>= 14
anx_long	double*	-	Target orbit ascending node crossing longitude	deg	[-180, 180]
drift_mode	long*	-	Flag to select between drift in mean local solar time and inclination as input characterization of the reference orbit	-	[0,1]
inclination	double*	-	If *drift_mode* = *XO_NOSUNSYNC_INCLINATION* Inclination of the reference orbit	deg	[0,180]
mlst_drift	double*	-	If *drift_mode* = *XO_NOSUNSYNC_DRIFT* Drift in mean local solar time of the reference orbit: · MLST[N+1]=MLST[N]+MLST drift	seconds/day	TBD

phase_increment	long*	-	If 1 then
			phase [N+1] = phase [N] + 1
			If 0 then
			phase [N+1] = phase [N]
				-	[0, 1]

output_dir	char*	-	Directory where the resulting OSF is written (if NULL, the cur rent directory is used)	-	-
output_filename	char*	-	Output OSF name if empty (i.e. “”), the software will generate the filename according to file name specification presented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output Orbit file	-	-
version_number	long*	-	Version number of output Orbit file	-	>= 1
fh_system	char*	-	System field of the output Orbit file fixed header	-	-

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

- Satellite ID: sat_id.
- Search direction.
- Drift mode: mlst_drift.
- Phase increment.

This CFI can append orbit changes for both sun-synchronous orbits and quasi-sun-synchronous orbits.

Use `drift_mode=XO_NOSUNSYNC_DRIFT` and `mlst_drift = 0.0` for a sun-synchronous orbit.

Use any other combination for the general case of quasi-sun-synchronous orbit.

7.39.4 Output parameters

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

Table 129: Output parameters of xo_gen_osf_change_repeat_cycle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
output_filename	char*	-	Name for output file. This is only an output parameter when it is empty (i.e. “”; see description of this parameter in Table 128)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.39.5 Warnings and errors

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

Table 130: Error messages of xo_gen_osf_change_repeat_cycle function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input values	Wrong value of one or more of the following input parameters: abs_orbit_number, search_direction, repeat_cycle, cycle_length, phase_increment Computation not performed	XO_CFI_GEN_OSF_CHANGE_INPUTS_ERR	0
ERR	Time ID is not initialized	Computation not performed	XO_CFI_GEN_OSF_CHANGE_TIME_INIT_ERR	1
ERR	Cannot read input OSF	Computation not performed	XO_CFI_GEN_OSF_CHANGE_READ_IN_OSF_ERR	2
ERR	Wrong drift mode	Wrong drift mode flag value for charaterization of non-sun-synchronous orbits Computation not performed	XO_CFI_GEN_OSF_CHANGE_DRIFT_MODE_ERR	3
ERR	Error calculating MLST drift	Error calculating MLST drift from inclination Computation not performed	XO_CFI_GEN_OSF_CHANGE_DRIFT_CALC_ERR	4
ERR	No transition found	No optimum transition found keeping orbit continuity Computation not performed	XO_CFI_GEN_OSF_CHANGE_NO_TRANSITION_ERR	5
ERR	Error calculating UTC of ANX	Error calculating the UTC time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_CHANGE_UTC_CALC_ERR	6
ERR	Error calculating TAI of ANX	Error calculating the TAI	XO_CFI_GEN_OSF_CHAN	7

		time of the orbit ascending node Computation not performed	GE_TAI_CALC_ERR	
ERR	Error calculating UT1 of ANX	Error calculating the UT1 time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_CHANGE_UT1_CALC_ERR	8
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_OSF_CHANGE_ALLOC_ERR	9
ERR	Error calculating the Fixed Header data	Computation not performed	XO_CFI_GEN_OSF_CHANGE_GET_FH_ERR	10
ERR	Error writing file to disk	Error writing the data structure to a file on disk Computation not performed	XO_CFI_GEN_OSF_CHANGE_WRITE_ERR	11

7.39.6 Runtime performances

The following runtime performance has been measured.

Table 131: Runtime performances of xo_gen_osf_change_repeat_cycle function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
24.0	11.2	13.6	6.7

7.39.7 Executable Program

The **gen_osf_change_repeat_cycle** executable program can be called from a Unix shell as:

gen_osf_change_repeat_cycle **-sat** satellite_name
-inosf input_filename
-orbit abs_orbit_number
[**-back**]
-repcyc repeat_cycle(days)
-cyclen cycle_length(orbits)
-anx anx_long(deg)
{ **-mlstdr** mlst_drift | **-inc** inclination }
[**-phinc**]
[**-dir** output_dir] (current directory by default)
[**-osf** output_filename] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
{ (**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) }) **}**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-back**] option for search_direction. Default value is xo_SEARCH_FORWARD. When the option is written, search_direction value is xo_SEARCH_BACKWARD.
- [**-phinc**] option for phase_increment. Default value is xo_NO_PHASE_INCREMENT. When the option is written, phase_increment value is xo_PHASE_INCREMENT.
- [**-xl_v**] option for EO_LIB Verbose mode.

- [**-xo_v**] option for EO_ORBIT 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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- The last three lines of parameters are used to initialize the time references. In order to do this, only one set of parameters should be introduced:
  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)

Example:

```
gen_osf_change_repeat_cycle -sat CRYOSAT
     -inosf CS_TEST_MPL_ORBREF_20020301T122001_99999999T999999_0001.EEF
     -orbit 400 -repcyc 369 -cyclen 5344 -anx 286.524398  -inc 92
     -dir ./gen_osf -osf mpl_orb_sc_at_304
     -tai -1100.1 -utc -1100.099595
     -ut1 -1100.0995914352 -gps -1100.0997801
```

7.40 xo_gen_osf_add_drift_cycle

7.40.1 Overview

Given a reference orbit from an existing OSF, a new requested orbit with a particular ascending node longitude and an orbit for the manoeuvre, the **xo_gen_osf_add_drift_cycle** CFI function fits a repeat cycle/ cycle length between the manoeuvre orbit (drift start) and the requested orbit (drift stop) such that the longitude of the ascending node at the drift stop orbit be the one requested.

The drift orbit is constrained by a maximum altitude difference with respect to the reference orbit.

Furthermore, if the reference orbit is sun-synchronous, the drift orbit shall also be sun-synchronous; but if the reference orbit is not sun-synchronous, the drift orbit shall keep the inclination constant.

This CFI appends two orbit changes to the existing OSF:

- The first one for the drift manoeuvre
- The second one for restoring the old reference orbit characteristics at the requested ascending node longitude

7.40.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long sat_id;
        xl_model_id model_id = {NULL};
        xl_time_id time_id = {NULL};
        long drift_start_orbit, drift_stop_orbit,
             phase_inc_start, phase_inc_stop, version_number;
        double drift_stop_anx_long, max_altitude_change;
        char input_filename[XD_MAX_STR],
             output_dir[XD_MAX_STR], output_filename[XD_MAX_STR];
        char *file_class, *fh_system;
        long status, ierr[XO_ERR_VECTOR_MAX_LENGTH];

        status = xo_gen_osf_add_drift_cycle (&sat_id, &model_id,
                                             &time_id,
                                             &input_filename,
                                             &drift_start_orbit,
                                             &drift_stop_orbit,
                                             &drift_stop_anx_long,
                                             &max_altitude_change,
```

```
                                        &phase_inc_start, &phase_inc_stop,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                        ierr);

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

        status = xo_gen_osf_add_drift_cycle_run (&run_id,
                                        &input_filename,
                                        &drift_start_orbit,
                                        &drift_stop_orbit,
                                        &drift_stop_anx_long,
                                        &max_altitude_change,
                                        &phase_inc_start, &phase_inc_stop,
                                        output_dir, output_filename,
                                        file_class, &version_number,
                                        fh_system,
                                        ierr);
    }
```

7.40.3 Input parameters

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

Table 132: Input parameters of xo_gen_osf_add_drift_cycle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
input_filename	char*	-	Input OSF to which the orbit changes are appended		
drift_start_orbit	long*	-	Absolute orbit number at the drift start	-	> abs orbit number in input OSF last orbit change
drift_stop_orbit	long*	-	Absolute orbit number at the drift stop	-	> drift_start_orbit
drift_stop_anx_long	double*	-	Drift stop orbit ascending node crossing longitude	deg	[-180, 180]
max_altitude_change	double*	-	Maximum variation in altitude between the reference orbit and the drift orbit	m	

phase_inc_start	long*	-	Phase increment at drift start
			If 1 then
			phase [N+1] = phase [N] + 1
			If 0 then
			phase [N+1] = phase [N]
				-	[0, 1]

phase_inc_stop	long*	-	Phase increment at drift stop
			If 1 then
			phase [N+1] = phase [N] + 1
			If 0 then
			phase [N+1] = phase [N]
				-	[0, 1]

output_dir	char*	-	Directory where the resulting OSF is written (if empty (i.e. ““), the current directory is used)	-	-
output_filename	char*	-	Output OSF name if empty (i.e. ““), the software will generate the filename according to file name specification pre sented in [FORMATS]. In such	-	-

			case, the generated name is returned in this variable		
file_class	char*	-	File class for output Orbit file	-	-
version_number	long*	-	Version number of output Orbit file	-	>= 1
fh_system	char*	-	System field of the output Orbit file fixed header	-	-

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

- Satellite ID: sat_id.
- Search direction.
- Drift mode: mlst_drift.
- Phase increment.

7.40.4 Output parameters

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

Table 133: Output parameters of xo_gen_osf_add_drift_cycle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
output_filename	char*	-	Name for output file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 132)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.40.5 Warnings and errors

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

Table 134: Error messages of xo_gen_osf_add_drift_cycle function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong input values	Wrong value of one or more of the following input parameters: drift_start_orbit, drift_stop_orbit, phase_inc_start, phase_inc_stop, Computation not performed	XO_CFI_GEN_OSF_DRIFT_INPUTS_ERR	0
ERR	Time ID is not initialized	Computation not performed	XO_CFI_GEN_OSF_DRIFT_TIME_INIT_ERR	1
ERR	Cannot read input OSF	Computation not performed	XO_CFI_GEN_OSF_DRIFT_READ_IN_OSF_ERR	2
ERR	No drift orbit necessary	Computation not performed	XO_CFI_GEN_OSF_DRIFT_NO_ADD_ERR	3
ERR	Error calculating inclination	Error calculating inclination for a no sun-synchronous orbit in order to keep inclination constant during the drift phase Computation not performed	XO_CFI_GEN_OSF_DRIFT_INCL_CALC_ERR	4
ERR	No drift orbit found	No drift orbit has been found that matches the drift start and stop ANX longitude Computation not performed	XO_CFI_GEN_OSF_DRIFT_NOT_FOUND_ERR	5
ERR	Error calculating UTC of ANX	Error calculating the UTC time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_DRIFT_UTC_CALC_ERR	6
ERR	Error calculating TAI of ANX	Error calculating the TAI time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_DRIFT_TAI_CALC_ERR	7
ERR	Error calculating UT1 of ANX	Error calculating the UT1 time of the orbit ascending node Computation not performed	XO_CFI_GEN_OSF_DRIFT_UT1_CALC_ERR	8
ERR	Memory allocation error	Computation not performed	XO_GEN_OSF_DRIFT_ALLOC_ERR	
ERR	Error calculating the Fixed Header data	Computation not performed	XO_GEN_OSF_DRIFT_GET_FH_ERR	

ERR	Error writing file to disk	Error writing the data structure to a file on disk Computation not performed	XO_CFI_GEN_OSF_DRIFT_WRITE_ERR	9

7.40.6 Runtime performances

The following runtime performance has been measured.

Table 135: Runtime performances of xo_gen_osf_add_drift_cycle function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
38.1	15.3	19.0	8.7

7.40.7 Executable Program

The **gen_osf_add_drift_cycle** executable program can be called from a Unix shell as:

gen_osf_add_drift_cycle **-sat** satellite_name
-inosf input_filename
-drorb0 drift_start_orbit
-drorb1 drift_stop_orbit
-anx drift_stop_anx_long (deg)
-alt max_altitude_change (m)
[**-phinc0**]
[**-phinc1**]
[**-dir** output_dir] (current directory by default)
[**-osf** output_filename] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
{ (**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) })**}**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-phinc0**] option for phase_inc_start. Default value is xo_NO_PHASE_INCREMENT. When the option is written, phase_inc_start value is xo_PHASE_INCREMENT.
- [**-phinc1**] option for phase_inc_stop. Default value is xo_NO_PHASE_INCREMENT. When the option is written, phase_inc_stop value is xo_PHASE_INCREMENT.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT 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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- The last three lines of parameters are used to initialize the time references. In order to do this, only one set of parameters should be introduced:
  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)

Example:

```
gen_osf_add_drift_cycle -sat CRYOSAT
-inosf CS_TEST_MPL_ORBREF_20020301T122001_99999999T999999_0001.EEF
-drorb0 30 -drorb1 2702 -anx 310 -alt 15000 -dir ./gen_osf
-osf mpl_orb_sc_at_305
-tai -1100.1 -utc -1100.099595 -ut1 -1100.0995914352 -gps -1100.0997801
```

7.41 xo_gen_rof

7.41.1 Overview

The **xo_gen_rof** CFI function creates a Restituted Orbit File (ROF) using as input one of the following reference file types:

- Orbit Scenario File
- FOS Predicted Orbit File
- DORIS Navigator File
- FOS Restituted Orbit File
- DORIS Preliminary Orbit File
- DORIS Precise Orbit FileTime of the ascending crossing node (TAI, UTC and UT1)
- The accepted output file types are:
- FOS Restituted Orbit File
- DORIS Preliminary Orbit File
- DORIS Precise Orbit FileTime

The time interval between consecutive OSVs can be selected by the user by means of a parameter in the calling interface. A flag for precise location of OSVs at “integer intervals” (e.g. every exact minute) is also available. If the reference file and the Restituted Orbit File contain OSVs at the same time, these OSVs will be identical.

Note: when using an OSF or Predicted Orbit file, the maximum time interval within the output Restituted orbit file is limited to 2 orbital periods before and after the middle point of the user requested time range.

7.41.2 Calling interface

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

```
#include <explorer_orbit.h>
{
       long sat_id;
       xl_model_id model_id = {NULL};
       xl_time_id time_id = {NULL};
       long   time_init, time_ref, start_orbit, stop_orbit,
              ref_filetype, rof_filetype, osv_precise, version_number;
       double start_time, stop_time, osv_interval;
       char   reference_file[XD_MAX_STR], output_dir[XD_MAX_STR],
              rof_filename[XD_MAX_STR], precise_conf_file[XD_MAX_STR];
       char  *file_class, *fh_system;
       long   status, ierr[XO_ERR_VECTOR_MAX_LENGTH];
```

```
        status = xo_gen_rof(&sat_id, &model_id, &time_id, &time_init,
                            &time_ref, &start_time,   &stop_time,
                            &start_orbit, &stop_orbit,
                            &osv_interval, &osv_precise,
                            &ref_filetype, reference_file,
                            precise_conf_file,
                            &rof_filetype, output_dir, rof_filename,
                            file_class, &version_number, fh_system,
                            /* output */
                            ierr);

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

        status = xo_gen_rof_run(&run_id, &time_init, &time_ref,
                                &start_time,   &stop_time,
                                &start_orbit, &stop_orbit,
                                &osv_interval, &osv_precise,
                                &ref_filetype, reference_file,
                                precise_conf_file,
                                &rof_filetype,
                                 output_dir, rof_filename,
                                file_class, &version_number, fh_system,
                                /* output */
                                ierr);
}
```

7.41.3 Input parameters

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

Table 136: Input parameters of xo_gen_rof function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations. NOTE: If time_id is not initialized, then time correlations will be initialized internally using the input reference file.	-	-
time_init	long*	-	Flag for selecting the time range of the initialisation.	-	Select either: · XO_SEL_ORBIT · XO_SEL_TIME
time_ref	long*	-	Time reference ID (see note in the ref_file type field)	-	Complete
start_time	double*	-	Processing time corresponding to the beginning of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
stop_time	double*	-	Processing time corresponding to the end of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
start_orbit	long*	-	Orbit number corresponding to the beginning of the required interval	orbits	>= 1
stop_orbit	long*	-	Orbit number corresponding to the end of the required interval	orbits	>= 1
osv_interval	double*	-	Interval between consecutive state vector. This parameter should be coherent with the osv_precise flag (see below). If osv_precise is set to: • xo_OSV_PRECISE_MINUTE: osv will be forced to be a multiple of 60 seconds. • xo_OSV_PRECISE_TEN_SECONDS: osv will be forced to be a multiple of	secs	>=0

			10 seconds.		
osv_precise	long*	-	Flag to indicate if state vectors should be placed at exact time locations	-	Complete
ref_filetype	long*	-	File type of the input reference file. (Note: When generating a ROF file from a DORIS NAVIGATOR file, the input times should be expressed in UTC)	-	Complete
reference_filename	char*	-	Reference File name	-	
precise_conf_file	char*	-	File with configuration parameters for precise propagator	-	If it is neither NULL nor "", precise propation will be used
rof_filetype	long*	-	File type of the output reference file	-	xo_REF_FILETYPE_ROF xo_REF_FILETYPE_DORIS_PREM xo_REF_FILETYPE_DORIS_PREC
output_dir	char*	-	Directory where the resulting ROF is written (if NULL, the current directory is used)	-	-
rof_filename	char*	-	Output ROF name if empty (i.e. ""), the software will generate the filename according to file name specification presented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output Restituted file	-	-
version_number	long*	-	Version number of output Restituted file	-	>= 1
fh_system	char*	-	System field of the output Restituted file fixed header	-	-

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

- Satellite ID: sat_id.
- Time initialisation: time_init.

- Time reference: time_ref.
- OSV precise: osv_precise. See this SUM.
- File type: ref_filetype and rof_filetype. See this SUM.

7.41.4 Output parameters

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

Table 137: Output parameters of xo_gen_rof function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
rof_filename	char*	-	Name for the output file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 136)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.41.5 Warnings and errors

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

Table 138: Error messages of xo_gen_rof function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite flag	Computation not performed	XO_CFI_GEN_ROF_WRONG_SAT_ID_ERR	0
ERR	Wrong input flag	Computation not performed	XO_CFI_GEN_ROF_WRONG_FLAG_ERR	1
ERR	Time ID is not initialized	Computation not performed	XO_CFI_GEN_ROF_TIME_I NIT_ERR	2
ERR	Could not initialise the time reference	Computation not performed	XO_CFI_GEN_ROF_TIME_I NITIALIZATION_ERR	3
ERR	Cannot initialise orbit ID	Computation not performed	XO_CFI_GEN_ROF_ORBIT_INIT_FILE_ERR	4
ERR	Cannot initialise the propagator	Computation not performed	XO_CFI_GEN_ROF_PROPAG_INIT_ERR	5

ERR	Could not perform a time <-> orbit transformation	Computation not performed	XO_CFI_GEN_ROF_TIME_ORBIT_ERR	6
ERR	Cannot initialise interpolation	Computation not performed	XO_CFI_GEN_ROF_INTERPOL_INIT_ERR	7
ERR	Cannot calculate state vector	Computation not performed	XO_CFI_GEN_ROF_CALCULATING_STATE_VECTOR _ERR	8
ERR	Cannot convert time to processing format	Computation not performed	XO_CFI_GEN_ROF_TIME_ERR	9
ERR	Cannot convert time from processing to external	Computation not performed	XO_CFI_GEN_ROF_TIME_TO_EXTERNAL_ERR	10
ERR	Cannot write ROF file to disk	Computation not performed	XO_CFI_GEN_ROF_WRITE _ERR	11
ERR	Error freeing memory	Computation not performed	XO_CFI_GEN_ROF_CLOSE_ERR	12
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_ROF_MEMORY_ERR	13
ERR	Error getting fixed header	Computation not performed	XO_CFI_GEN_ROF_GET_F H_ERR	14
ERR	OSV interval is not compatible with OSV Precise flag. The OSV Interval will be set to %f seconds.	Computation performed with a different value for the osv_interval	XO_CFI_GEN_ROF_WRONG_INTERVAL_WARN	15
ERR	Error reading precise propagator configuration file	Computation not performed	XO_CFI_GEN_ROF_READ_PRECISE_FILE_ERR	16

7.41.6 Runtime performances

The following runtime performance has been measured.

Table 139: Runtime performances of xo_gen_rof function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
522.6	213.9	252.7	75.9

7.41.7 Executable Program

The **gen_rof** executable program can be called from a Unix shell as:

gen_rof **-sat** satellite_name
-tref time_ref
{ **-tstart** start_time **-tstop** stop_time (decimal days) |
-tastart start_time **-tastop** stop_time (CCSDSA format) |
-ostart start_orbit **-ostop** stop_orbit (orbits) }
-osvint osv_interval
[**-osvpre**]
-reftyp ref_file_type
-ref reference_file
-roftyp rof_file_type
[**-precfile** precise_conf_file] (empty string by default)
[**-dir** output_dir] (current directory by default)
[**-rof** output_filename] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version= 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
[(**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) }) **]**

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-osvpre**] option for osv_precise.Default value is xo_OSV_PRECISE_NO. When the option is written, ovs_precise value is xo_OSV_PRECISE_MINUTE.
- [**-xl_v**] option for EO_LIB Verbose mode.

- [**-xo_v**] option for EO_ORBIT 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, SENTINEL_1A, SENTINEL_1B, SENTINEL_2, SENTINEL_3, SEOSAT, GENERIC.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *ref_file_type*: OSF, POF, DORISNAV, ROF, DORISPREM, DORISPREC.
- Possible values for *rof_file_type*: ROF, DORISPREM, DORISPREC.
- Possible values for *time_ref* and *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- Time references need to be initialized only when using OSF as the type of the input reference file. The inputs needed for this issue are provided in the last three lines of parameters. Note that only one set of parameters should be introduced:
  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)
- Precise propagation is used if precfile is provided.

Example:

```
gen_rof   -sat CRYOSAT -tref TAI -ostart 1000 -ostop 1001
          -osvint 300  -reftyp OSF
          -ref
            CS_TEST_MPL_ORBREF_20020301T122001_99999999T999999_0001.EEF
          -roftyp ROF -dir ./gen_rof/ -rof orb_res_file_at_306
          -tmod FOS_PREDICTED -tfile ./data/test.fpo -trid TAI
          -tm0 0 -tm1 10000
```

7.42 xo_gen_rof_prototype

7.42.1 Overview

The **xo_gen_rof_prototype** CFI function creates a Restituted Orbit File (ROF) using the following input parameters:

- Date (processing time) and orbit
- Longitude of the ascending node,
- Satellite Repeat Cycle and Cycle Length
- Mean local solar time at ascending node
- Drift of mean local solar time or the inclination

The time interval between consecutive OSVs can be selected by the user by means of a parameter in the calling interface.

A file with the configuration parameters for precise propagator can be introduced. In this case, the numeric propagator is used.

7.42.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long sat_id;
        xl_model_id model_id = {NULL};
        xl_time_id time_id = {NULL};
        long propag_model, time_ref, time_init_mode;
        long orbit0, drift_mode, irep, icyc, start_orbit, stop_orbit;
        double time0, start_time, stop_orbit, osv_interval;
        double ascmlst_drift, inclination, rlong, ascmlst;
        char   output_dir[XD_MAX_STR], rof_filename[XD_MAX_STR];
        char  *file_class, *fh_system;
        long status, ierr[XO_ERR_VECTOR_MAX_LENGTH], version_number;

        status = xo_gen_rof_prototype (&sat_id, &model_id, &time_id,
                                       &propag_model, &time_ref,
                                       &time0, &orbit0, &time_init_mode,
                                       &start_time, &start_orbit
                                       &stop_time, &stop_orbit,
                                       &drift_mode,
```

```
                                        &ascmlst_drift, &inclination,
                                        &irep, &icyc, &rlong, &ascmlst,
                                        &osv_interval
                                        output_dir,      rof_filename,
                                        file_class, &version_number,
                                        fh_system,
                                       /* output */
                                       ierr);
        /* Or, using the run_id */
        long run_id;

        status = xo_gen_rof_prototype_run (&run_id,
                                        &propag_model, &time_ref,
                                        &time0, &orbit0, &time_init_mode,
                                        &start_time, &start_orbit
                                        &stop_time, &stop_orbit,
                                        &drift_mode,
                                        &ascmlst_drift, &inclination,
                                        &irep, &icyc, &rlong, &ascmlst,
                                        &osv_interval
                                        output_dir,      rof_filename,
                                        file_class, &version_number,
                                        fh_system,
                                       /* output */
                                       ierr);
    }
```

7.42.3 Input parameters

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

Table 140: Input parameters of xo_gen_rof_prototype function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
propag_model	long*	-	Propagation model ID	-	Complete
time_ref	long*	-	Time reference ID	-	Complete
time0	double*	-	Reference time	Decimal days (Processing format)	[-18262.0,36524.0]
orbit0	long*	-	Absolute orbit number of the reference orbit	-	>= 0
time_init_mode	long*	-	Flag for selecting the time range of the initialisation.	-	Select either: · XO_SEL_ORBIT · XO_SEL_TIME
start_time	double*	-	Processing time corresponding to the beginning of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
start_orbit	long*	-	Orbit number corresponding to the beginning of the required interval	orbits	>= 1
stop_time	double*	-	Processing time corresponding to the end of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
stop_orbit	long*	-	Orbit number corresponding to the end of the required interval	orbits	>= 1
drift_mode	long*	-	Flag to select between drift in mean local solar time and inclination as input characterization of the reference orbit	-	Complete
ascmlst_drift	double*	-	If *drift_mode* = *XO_NOSUNSYNC_MLST* Drift in mean local solar time of the reference orbit	seconds/day	TBD
inclination	double*	-	If *drift_mode* = *XO_NOSUNSYNC_INCLINATIO*	deg	[0,180]

			N Inclination of the reference orbit		
irep	long *	-	Repeat cycle of the reference orbit The actual repeat cycle is calcu lated as per definition included in .	days	> 0
icyc	long *	-	Cycle length of the reference orbit	orbits	> 0
rlong	double*	-	Geocentric longitude of the [Earth fixed] ascending node (Earth fixed CS)	deg	[0,360)
ascmlst	double*	-	Mean local solar time at ascending node	hours	[0, 24)
osv_interval	double*	-	Interval between consecutive state vector	secs	>=0
output_dir	char*	-	Directory where the resulting ROF is written (if NULL, the cur rent directory is used)	-	-
rof_filename	char*	-	Output ROF name if empty (i.e. ““), the software will generate the filename according to file name specification pre sented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output Restituted file	-	-
version_number	long*	-	Version number of output Restituted file	-	>= 1
fh_system	char*	-	System field of the output Restituted file fixed header	-	-

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

- Satellite ID: sat_id.
- Time initialisation: time_init.
- Time reference: time_ref.
- Drift Mode: drift_mode.

7.42.4 Output parameters

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

Table 141: Output parameters of xo_gen_rof_prototype function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
rof_filename	char*	-	Name for the output file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 144)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.42.5 Warnings and errors

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

Table 142: Error messages of xo_gen_rof_prototype function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite flag	Computation not performed	XO_CFI_GEN_ROF_PROTOTYPE_WRONG_SAT_ID_ERR	0
ERR	Time ID is not initialized	Computation not performed	XO_CFI_GEN_ROF_PROTOTYPE_TIME_ID_ERR	1
ERR	Wrong input flag	Computation not performed	XO_CFI_GEN_ROF_PROTOTYPE_WRONG_FLAG_ERR	2
ERR	Cannot initialise propagator	Computation not performed	XO_CFI_GEN_ROF_PROTOTYPE_PROPAG_INIT_DEF_ERR	3
ERR	Cannot calculate state vector	Computation not performed	XO_CFI_GEN_ROF_PROTOTYPE_CALCULATIN	3

			G_STATE_VECTOR_ER R	
ERR	Cannot convert time in processing reference	Computation not performed	XO_CFI_GEN_ROF_PRO TOTYPE_TIME_ERR	5
ERR	Cannot convert time from processing to external	Computation not performed	XO_CFI_GEN_ROF_PRO TOTYPE_TIME_TO_EXT ERNAL_ERR	6
ERR	Error freeing memory	Computation not performed	XO_CFI_GEN_ROF_PRO TOTYPE_CLOSE_ERR	7
ERR	Error creating the fixed header	Computation not performed	XO_CFI_GEN_ROF_PROT OTYPE_GET_FH_ERR	8
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_ROF_PROT OTYPE_MEMORY_ERR	9
ERR	Cannot write ROF XML file	Computation not performed	XO_CFI_GEN_ROF_PRO TOTYPE_WRITE_ERR	10

7.42.6 Runtime performances

The following runtime performance has been measured.

Table 143: Runtime performances of xo_gen_rof_prototype function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TDB	TDB	TDB	TDB

7.43 xo_gen_pof

7.43.1 Overview

The **xo_gen_pof** CFI function creates a Predicted Orbit File (POF) with one state vector per orbit using as input one of the following reference file types:

- Orbit Scenario File
- FOS Predicted Orbit File
- DORIS Navigator File
- FOS Restituted Orbit File
- DORIS Preliminary Orbit File
- DORIS Precise Orbit FileTime of the ascending crossing node (TAI, UTC and UT1)

The location of the state vector within the orbit can be selected by the user by means of a parameter in the calling interface. If the reference file and the Predicted Orbit File contain OSVs at the same time, these OSVs will be identical.

A file with the configuration parameters for precise propagator can be introduced. In this case, the numeric propagator is used.

7.43.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long    sat_id;
        xl_model_id model_id = {NULL};
        xl_time_id time_id = {NULL};
        long    time_init, time_ref, start_orbit, stop_orbit,
               ref_filetype, pof_filetype, version_number;
        double start_time, stop_time, osv_location;
        char    reference_file[XD_MAX_STR], output_dir[XD_MAX_STR],
               pof_filename[XD_MAX_STR], precise_conf_file[XD_MAX_STR];
        char   *file_class, *fh_system;
        long    status, ierr[XO_ERR_VECTOR_MAX_LENGTH];
        status = xo_gen_pof(&sat_id, &model_id, &time_id,
                            &time_init, &time_ref,
                            &start_time, &stop_time,
                            &start_orbit, &stop_orbit, &osv_location,
                            &ref_filetype, reference_file,
                            precise_conf_file,
```

```
                        &pof_filetype, output_dir,
        pof_filename,
                        file_class, &version_number, fh_system,
                         /* output */
                        ierr);


        /* Or, using the run_id */
        long run_id;
        status = xo_gen_pof_run(&run_id,
                                &time_init, &time_ref,
                                &start_time,  &stop_time,
                                &start_orbit, &stop_orbit,
                                &osv_location,
                                &ref_filetype, reference_file,
                                precise_conf_file,
                               &pof_filetype, output_dir,
                                pof_filename,
                                file_class, &version_number, fh_system,
                               /* output */
                               ierr);
}
```

7.43.3 Input parameters

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

Table 144: Input parameters of xo_gen_pof function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id	-	Model ID	-	Complete
time_id	xl_time_id*	-	Structure that contains the time correlations. NOTE: If time_id is not initialized, then time correlations will be initialized internally using the input reference file	-	-
time_init	long*	-	Flag for selecting the time range of the initialisation.	-	Select either: · XO_SEL_ORBIT · XO_SEL_TIME
time_ref	long*	-	Time reference ID. (See note in the ref_filetype field)	-	Complete
start_time	double*	-	Processing time corresponding to the beginning of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
stop_time	double*	-	Processing time corresponding to the end of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
start_orbit	long*	-	Orbit number corresponding to the beginning of the required interval	orbits	>= 1
stop_orbit	long*	-	Orbit number corresponding to the end of the required interval	orbits	>= 1
osv_location	double*	-	Location of the state vector within the orbit	secs	>=0 < 1 nodal period
ref_filetype	long*	-	File type of the input reference file. (Note: When generating a POF file from a DORIS NAVIGATOR file, the input times should be expressed in UTC)	-	Complete
reference_filename	char*	-	Reference File name	-	
precise_conf_file	char*	-	File with precise propagator con figuration	-	If it is not neither NULL nor "",

					precise propagation will be used
pof_filetype	long*	-	File type of the output reference file	-	XO_REF_FILETYPE_POF
output_dir	char*	-	Directory where the resulting POF is written (if NULL, the current directory is used)	-	-
pof_filename	char*	-	Output POF name if empty (i.e. ""), the software will generate the filename according to file name specification presented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output Predicted file	-	-
version_number	long*	-	Version number of output Predicted file	-	>= 1
fh_system	char*	-	System field of the output Predicted file fixed header	-	-

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

- Satellite ID: sat_id.
- Time initialisation: time_init.
- Time reference: time_ref.
- File type: ref_filetype and pof_filetype. See section 6.2 in this SUM.

7.43.4 Output parameters

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

Table 145: Output parameters of xo_gen_pof function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
pof_filename	char*	-	Name for the output file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 140)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.43.5 Warnings and errors

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

Table 146: Error messages of xo_gen_pof function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite flag	Computation not performed	XO_CFI_GEN_POF_WRONG_SAT_ID_ERR	0
ERR	Wrong input flag	Computation not performed	XO_CFI_GEN_POF_WRONG_FLAG_ERR	1
ERR	Time ID is not initialized	Computation not performed	XO_CFI_GEN_POF_TIME_INIT_ERR	2
ERR	Could not initialise the time reference	Computation not performed	XO_CFI_GEN_POF_TIME_INITIALIZATION_ER R	3
ERR	Cannot initialise orbit	Computation not performed	XO_CFI_GEN_POF_ORBIT_INIT_FILE_ERR	4
ERR	Cannot initialise propagation	Computation not performed	XO_CFI_GEN_POF_PROPAG_INIT_ERR	5
ERR	Cannot initialise interpolation	Computation not performed	XO_CFI_GEN_POF_INTERPOL_INIT_ERR	6
ERR	Wrong interpol initialisation	Computation not performed	XO_CFI_GEN_POF_INTERNAL1_ERR	7
ERR	Cannot calculate state vector	Computation not performed	XO_CFI_GEN_POF_CALCULATING_STATE_VECTOR_ERR	8
ERR	Error freeing memory	Computation not performed	XO_CFI_GEN_POF_CLOSE_ERR	9
ERR	Time transformation error	Computation not performed	XO_CFI_GEN_POF_TIME_TRANS_ERR	10
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_POF_MEMORY_ERR	11
ERR	Error creating the fixed header	Computation not performed	XO_CFI_GEN_POF_GET_FH_ERR	12
ERR	Error writing POF file to disk	Computation not performed	XO_CFI_GEN_POF_WRITE_ERR	13
ERR	Error reading configuration	Computation not performed	XO_CFI_GEN_POF_READ	14

	file for precise propagation		_ PRECISE_FILE_ERR	
ERR	Error converting time to orbit or orbit to time	Computation not performed	XO_CFI_GEN_POF_ORBIT _TIME_ERR	15

7.43.6 Runtime performances

The following runtime performance has been measured.

Table 147: Runtime performances of xo_gen_pof function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
70.5	26.2	32.1	11.5

7.43.7 Executable Program

The **gen_pof** executable program can be called from a Unix shell as:

gen_pof **-sat** satellite_name
-tref time_ref
{ **-tstart** start_time **-tstop** stop_time (decimal days) |
-tastart start_time **-tastop** stop_time (CCSDSA format) |
-ostart start_orbit **-ostop** stop_orbit (orbits) }
-osvloc osv_location (secs)
-reftyp ref_file_type
-ref reference_file
-poftyp pof_file_type
[**-precfile** precise_conf_file] (empty string by default)
[**-dir** output_dir] (current directory by default)
[**-pof** output_filename] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
[(**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) })]

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT 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, SENTINEL_1A, SENTINEL_1B,SENTINEL_2, SENTINEL_3, SEOSAT, GENERIC.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *ref_file_type* and *pof_file_type*: OSF, POF, DORISNAV, ROF, DORISPREM, DORISPREC.
- Possible values for *time_ref* and *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- Time references need to be initialized only when using OSF as the type of the input reference file. The inputs needed for this issue are provided in the last three lines of parameters. Note that only one set of parameters should be introduced:
  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)
- Precise propagation is used if precfile is provided.

Example:

```
gen_pof -sat CRYOSAT -tref GPS -ostart 13 -ostop 14 -osvloc 0 -reftyp OSF
     -ref CS_TEST_MPL_ORBREF_20020301T122001_99999999T999999_0001.EEF
     -poftyp POF -dir ./gen_pof/ -pof orb_pre_file_at_307
          -tai -1100.1 -utc -1100.099595 -ut1 -1100.0995914352
          -gps -1100.0997801
```

7.44 xo_gen_oef

7.44.1 Overview

The **xo_gen_oef** CFI function creates an Orbit Event by merging an Orbit Scenario file (OSF) and a Predicted Orbit File.

7.44.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        char    oef[XD_MAX_STR], osf[XD_MAX_STR],
                pof[XD_MAX_STR];
        char   *file_class, *fh_system;
        long version_number;
        long    status, ierr[XO_NUM_ERR_GEN_OEF];
        status = xo_gen_oef(&oef, &osf, &pof,
                            file_class, &version_number, fh_system,
                           /* output */
                           ierr);
}
```

7.44.3 Input parameters

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

Table 148: Input parameters of xo_gen_oef function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
oef	char[]	-	Output OEF name. If empty (i.e. ""), the software will generate the filename according to file name specification presented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
osf	char*	-	Orbit Scenario File name	-	-
pof	char*	-	Predicted Orbit File name	-	-
file_class	char*	-	File class for output file (dummy in the current version)	-	-
version_number	long*	-	Version number of output file (dummy	-	>= 1

			in the current version)		
fh_system	char*	-	System field of the output file fixed header (dummy in the current version)	-	-

7.44.4 Output parameters

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

Table 149: Output parameters of xo_gen_oef function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_gen_oef	long	-	Main status flag	-	-1, 0, +1
oef	char*	-	Name for the output file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 148)	-	-
ierr[]	long	all	Status vector	-	-

7.44.5 Warnings and errors

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

Table 150: Error messages of xo_gen_oef function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Could not open output file for writing	Computation not performed	XO_CFI_GEN_OEF_OPEN_FILE_ERR	0
ERR	Could not copy the Orbit Scenario file	Computation not performed	XO_CFI_GEN_OEF_COPY_FILE_ERR	1
ERR	Could not copy "List_of_OSVs" in the output file	Computation not performed	XO_CFI_GEN_OEF_COPY_NODE_ERR	2
ERR	Could not close output file	Computation not performed	XO_CFI_GEN_OEF_CLOS	3

			E_ERR	
ERR	Error reading the fixed header from the Orbit Scenario file	Computation not performed	XO_CFI_GEN_OEF_READ_ OSF_ERR	4
ERR	Error reading the fixed header from the Predicted Orbit file	Computation not performed	XO_CFI_GEN_OEF_READ_ POF_ERR	5
ERR	Could not write the Orbit Event file	Computation not performed	XO_CFI_GEN_OEF_WRITE_ERR	6
ERR	Could not get the current time	Computation not performed	XO_CFI_GEN_OEF_CURRENT_TIME_ERR	7
WARN	Cannot write schema in the file	Computation performed. The output file does not contain the schema reference in the root tag	XO_CFI_GEN_OEF_SET_SCHEMA_WARN	8

7.44.6 Runtime performances

The following runtime performance has been measured.

Table 151: Runtime performances of xo_gen_oef function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
42.3	22.2	25.3	10.9

7.44.7 Executable Program

The **gen_oef** executable program can be called from a Unix shell as:

gen_oef **-osf** name of the orbit scenario file
-pof
[**-oef**] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1 by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xd_v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [-xl_v] option for EO_LIB Verbose mode.
- [-xo_v] option for EO_ORBIT Verbose mode.
- [-v] option for Verbose mode for all libraries (default is Silent).
- [-show] displays the inputs of the function and the results.

Example:

```
gen_oef -osf ./input_osf.xml -pof ./input_pof.xml
        -flcl OPER -vers 0 -show -v
```

7.45 xo_gen_dnf

7.45.1 Overview

The **xo_gen_dnf** CFI function creates a DORIS Navigator File using as input one of the following reference file types:

- Orbit Scenario File
- FOS Predicted Orbit File
- FOS Restituted Orbit File
- DORIS Navigator File
- DORIS Preliminary Orbit File
- DORIS Precise Orbit FileTime of the ascending crossing node (TAI, UTC and UT1)
- The accepted output file types are:
- FOS Restituted Orbit File
- DORIS Preliminary Orbit File
- DORIS Precise Orbit FileTime

The time interval between consecutive OSVs can be selected by the user by means of a parameter in the calling interface. A flag for precise location of OSVs at “integer intervals” (e.g. every exact minute or every ten seconds) is also available. If the reference file and the DORIS Navigator File contain OSVs at the same time, these OSVs will be identical.

An optional control file can be introduced to correct the state vectors. This file contains the corrections for position and velocity in the along, across and radial directions. The format of this file is shown in [D_H_SUM].

A file with the configuration parameters for precise propagator can be introduced. In this case, the numeric propagator is used.

Note: when using an OSF or Predicted Orbit file, the maximum time interval within the output Doris Navigator file is limited to 2 orbital periods before and after the middle point of the user requested time range.

7.45.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long    sat_id;
        xl_time_id time_id = {NULL};
        xl_model_id model_id = {NULL};
        long    time_init, time_ref, start_orbit, stop_orbit,
```

```
            ref_filetype, dnf_filetype, osv_precise, version_number;
        double start_time, stop_time, osv_interval;
        char   reference_file[XD_MAX_STR], output_dir[XD_MAX_STR],
               dnf_filename[XD_MAX_STR], ctrl_file[XD_MAX_STR],
               precise_conf_file[XD_MAX_STR];
        char  *file_class, *fh_system;
        long   status, ierr[XO_ERR_VECTOR_MAX_LENGTH];

        status = xo_gen_dnf(&sat_id, &model_id, &time_id,
                            &time_init, &time_ref,
                            &start_time, &stop_time,
                            &start_orbit, &stop_orbit,
                            &osv_interval, &osv_precise,
                            &ref_filetype, reference_file, ctrl_file,
                            precise_conf_file,
                            &dnf_filetype, output_dir, dnf_filename,
                            file_class, &version_number, fh_system,
                            /* output */
                            ierr);

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

        status = xo_gen_dnf_run(&run_id,
                                &time_init, &time_ref,
                                &start_time,  &stop_time,
                                &start_orbit, &stop_orbit,
                                &osv_interval, &osv_precise,
                                &ref_filetype, reference_file,ctrl_file,
                                precise_conf_file,
                                &dnf_filetype, output_dir, dnf_filename,
                                file_class, &version_number, fh_system,
                                /* output */
                                ierr);
}
```

7.45.3 Input parameters

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

Table 152: Input parameters of xo_gen_dnf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations. NOTE: If time_id is not initialized, then time correlations will be initialized internally using the input reference file	-	-
time_init	long*	-	Flag for selecting the time range of the initialisation.	-	Select either: · XO_SEL_ORBIT · XO_SEL_TIME
time_ref	long*	-	Time reference ID (see note in the ref_filetype field)	-	Complete
start_time	double*	-	Processing time corresponding to the beginning of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
stop_time	double*	-	Processing time corresponding to the end of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
start_orbit	long*	-	Orbit number corresponding to the beginning of the required interval	orbits	>= 1
stop_orbit	long*	-	Orbit number corresponding to the end of the required interval	orbits	>= 1
osv_interval	double*	-	Interval between consecutive state vector. This parameter should be coherent with the osv_precise flag (see below). If osv_precise is set to: • xo_OSV_PRECISE_MINUTE: osv will be forced to be a multiple of 60 seconds. • xo_OSV_PRECISE_TEN_SECONDS: osv will be forced to be a multiple of	secs	>=0

			10 seconds.		
osv_precise	long*	-	Flag to indicate if state vectors should be placed at exact time locations	-	Complete
ref_filetype	long*	-	File type of the input reference file. (Note: When generating a DNF file from another DORIS NAVIGA TOR file, the input times should be expressed in UTC)	-	Complete
reference_filename	char*	-	Reference File name	-	
ctrl_file	char*	-	Control File in xml format. This file contains the corrections for position and velocity in the along, across and radial directions together with the position accuracy(see [D_H_SUM].) If empty string (““), no corrections will be performed and the accu racy (quality index in the DNF)will be set to 1.	-	-
precise_conf_file	char*	-	File with precise propagator con figuration	-	If it is not neither NULL nor ““, precise propagation will be used
dnf_filetype	long*	-	File type of the output DORIS Navigator file	-	xo_REF_FILETYPE_DORIS_NAV
output_dir	char*	-	Directory where the resulting DNF is written (if NULL, the cur rent directory is used)	-	-
dnf_filename	char*	-	Output DNF name if empty (i.e. ““), the software will generate the filename according to file name specification pre sented in [FORMATS]. In such case, the generated name is returned in this variable	-	-
file_class	char*	-	File class for output file (dummy in the current version)	-	-
version_number	long*	-	Version number of output file (dummy in the current version)	-	>= 1
fh_system	char*	-	System field of the output file fixed header (dummy in the current version)	-	-

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

- Satellite ID: sat_id.
- Time initialisation: time_init.
- Time reference: time_ref.
- OSV precise: osv_precise. See this SUM.
- File type: ref_filetype and rof_filetype. See this SUM.

7.45.4 Output parameters

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

Table 153: Output parameters of xo_gen_dnf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
dnf_filename	char*	-	Name for the output file. This is only an output parameter when it is empty (i.e. ""; see description of this parameter in Table 152)	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.45.5 Warnings and errors

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

Table 154: Error messages of xo_gen_dnf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite flag	Computation not performed	XO_CFI_GEN_DNF_WRONG_SAT_ID_ERR	0
ERR	Wrong input flag	Computation not performed	XO_CFI_GEN_DNF_WRONG_FLAG_ERR	1
ERR	Time ID is not initialized	Computation not performed	XO_CFI_GEN_DNF_TIME_INIT_ERR	2
ERR	Could not initialise the time reference	Computation not performed	XO_CFI_GEN_DNF_TIME_INITIALIZATION_ERR	3
ERR	Cannot initialise orbit ID	Computation not performed	XO_CFI_GEN_DNF_ORBIT_INIT_FILE_ERR	4
ERR	Cannot initialise the propagator	Computation not performed	XO_CFI_GEN_DNF_PROPAG_INIT_ERR	5
ERR	Cannot initialise interpolation	Computation not performed	XO_CFI_GEN_DNF_INTERPOL_INIT_ERR	6
ERR	Could not perform a time <-> orbit transformation	Computation not performed	XO_CFI_GEN_DNF_TIME_ORBIT_ERR	7
ERR	Error in a time transformation function	Computation not performed	XO_CFI_GEN_DNF_TIME_ERR	8
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_DNF_MEMORY_ERR	9
ERR	Cannot calculate state vector	Computation not performed	XO_CFI_GEN_DNF_CALCULATING_STATE_VECTOR _ERR	10
ERR	Error reading the Control File	Computation not performed	XO_CFI_GEN_DNF_READ_ CONTROL_FILE_ERR	11
ERR	Cannot correct state vector	Computation not performed	XO_CFI_GEN_DNF_CORRECT_OSV_ERR	12
ERR	Error changing state vector	Computation not performed	XO_CFI_GEN_DNF_CHAN	13

	from EF to J2000		GE_COORD_ERR	
ERR	Error creating the DORIS header	Computation not performed	XO_CFI_GEN_DNF_COMPUTE_HEADER_ERR	14
ERR	Error freeing memory	Computation not performed	XO_CFI_GEN_DNF_CLOSE_ERR	15
ERR	Cannot write DORIS Data Block file	Computation not performed	XO_CFI_GEN_DNF_WRITE_FILE_ERR	16
WARN	OSV interval is not compatible with OSV Precise flag. The OSV Interval will be set to %f seconds.	Computation performed with a different value for the osv_interval	XO_CFI_GEN_DNF_WRONG_INTERVAL_WARN	17
ERR	Error reading precise propagator configuration file	Computation not performed	XO_CFI_GEN_DNF_READ_ PRECISE_FILE_ERR	18

7.45.6 Runtime performances

The following runtime performance has been measured.

Table 155: Runtime performances of xo_gen_dnf function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
4246.0	1895.0	1936.0	353.0

Executable Program

The **gen_dnf** executable program can be called from a Unix shell as:

gen_dnf **-sat** satellite_name
-tref time_ref
{ -tstart start_time -tstop stop_time (decimal days) |
-tastart start_time -tastop stop_time (CCSDSA format) |
-ostart start_orbit -ostop stop_orbit (orbits) }
-osvint osv_interval
[**-osvpre**]
-reftyp ref_file_type
-ref reference_file
[**-ctrl** control_file]
[**-precfile** precise_conf_file] (empty string by default)
[**-dir** output_dir] (current directory by default)
[**-dnf** output_filename] (default: name generated automatically)
[**-flcl** file_class] (empty string by default)
[**-vers** version] (version = 1by default)
[**-fhsys** fh_system] (empty string by default)
[**-v**]
[**-xl_v**]
[**-xo_v**]
[**-help**]
[**-show**]
[(**-tai** TAI_time **-gps** GPS_time **-utc** UTC_time **-ut1** UT1_time) |
(**-tmod** time_model **-tfile** time_file **-trid** time_reference
{(**-tm0** time0 **-tm1** time1) | (**-orb0** orbit0 **-orb1** orbit1) })]

Note that:

- Order of parameters does not matter.
- Bracketed parameters are not mandatory.
- Options between curly brackets and separated by a vertical bar are mutually exclusive.
- [**-osvpre**] option for osv_precise.Default value is xo_OSV_PRECISE_NO. When the option is written, ovs_precise value is xo_OSV_PRECISE_MINUTE.
- [**-xl_v**] option for EO_LIB Verbose mode.
- [**-xo_v**] option for EO_ORBIT 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.
- Possible values for *time_model*: USER, NONE , IERS_B_PREDICTED, IERS_B_RESTITUTED, FOS_PREDICTED, FOS_RESTITUTED, DORIS_PRELIMINARY, DORIS_PRECISE, DORIS_NAVIGATOR.
- Possible values for *ref_file_type*: OSF, POF, DORISNAV, ROF, DORISPREM, DORISPREC.
- Possible values for *time_ref* and *time_reference*: UNDEF, TAI, UTC, UT1, GPS.
- Time references need to be initialized only when using OSF as the type of the input reference file. The inputs needed for this issue are provided in the last three lines of parameters. Note that only one set of parameters should be introduced:
  - TAI, GPS, UTC and UT1 input times (as in xl_time_ref_init)
  - A file with time reference data, the time mode, the time reference name and a time range (as in xl_time_ref_init_file)
- Precise propagation is used if precfile is provided.

Example:

```
gen_dnf -sat CRYOSAT -tref UTC -tstart 0.99650462962963
        -tstop 01386574074708 -osvint 20 -reftyp ROF
        -ref EARTH_EXPLORER_FRO_TO_DORIS_2000
        -ctrl CONTROL_FILE.xml -dir ./gen_dnf/ -dnf doris_nav_at_308
        -tai 0.000000 -utc -4.0509259e-4 -ut1 -4.1435185185e-4
        -gps 2.1991e-4 -show
```

7.46 xo_gen_tle

7.46.1 Overview

The **xo_gen_tle** CFI function creates TLE File using as input a Predicted Orbit File. It is possible to select the way in which the TLE are generated:

- Generate a TLE per OSV in the orbit file (XO_ONE_TLE_PER_OSV).
- Find the best TLE which fits to the OSVs in the orbit file (XO_FIT_TLE).

7.46.2 Calling interface

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

```
#include <explorer_orbit.h>
{
       long   sat_id;
       xl_time_id time_id = {NULL};
       long   fit_mode, time_mode, time_ref, start_orbit, stop_orbit;
       double start_time, stop_time;
       char   reference_file[XD_MAX_STR], tle_filename[XD_MAX_STR];
       long   status, ierr[XO_ERR_VECTOR_MAX_LENGTH];

       status = xo_gen_tle (&sat_id, &fit_mode,
                            &time_mode, &time_ref,
                            &start_time, &stop_time,
                            &start_orbit, &stop_orbit,
                            reference_file, tle_filename,
                            ierr);
}
```

7.46.3 Input parameters

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

Table 156: Input parameters of xo_gen_tle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
fit_mode	long*	-	fitting mode	-	Complete
time_mode	long*	-	Flag for selecting the time range of the initialisation.	-	Select either: • XO_SEL_ORBIT • XO_SEL_TIME • XO_SEL_DEFAULT
time_ref	long*	-	Time reference for the input start_time and stop_time	-	Complete
start_time	double*	-	Processing time corresponding to the beginning of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
stop_time	double*	-	Processing time corresponding to the end of the required interval	Decimal days, MJD2000	[-18262.0,36524.0]
start_orbit	long*	-	Orbit number corresponding to the beginning of the required interval	orbits	>= 1
stop_orbit	long*	-	Orbit number corresponding to the end of the required interval	orbits	>= 1
reference_file	char*	-	Reference File name	-	-

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

- Fitting mode: fit_mode. See this SUM.
- Satellite ID: sat_id.
- Time initialisation: time_mode.
- Time reference: time_ref.

7.46.4 Output parameters

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

Table 157: Output parameters of xo_gen_tle function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
tle_filename	char*	-	Name for the output file.	-	-
ierr[XO_ERR_VECTOR_MAX_LENGTH]	long	all	Status vector	-	-

7.46.5 Warnings and errors

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

Table 158: Error messages of xo_gen_tle function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Wrong satellite ID.	Computation not performed	XO_CFI_GEN_TLE_WRONG_SAT_ID_ERR	0
ERR	Wrong input time reference	Computation not performed	XO_CFI_GEN_TLE_WRONG_FLAG_ERR	1
ERR	Wrong input fitting mode	Computation not performed	XO_CFI_GEN_TLE_WRONG_FIT_MODE_ERR	2
ERR	Could not initialise the time correlations from input file: %s	Computation not performed	XO_CFI_GEN_TLE_TIME_INITIALIZATION_ERR	3
ERR	Could not initialise the orbit data from input file: %s	Computation not performed	XO_CFI_GEN_TLE_ORBIT_INIT_FILE_ERR	4
ERR	Memory allocation error	Computation not performed	XO_CFI_GEN_TLE_MEMORY_ERR	5
ERR	Could not generate the TLE for orbit %ld	Computation not performed	XO_CFI_GEN_TLE_OSV_TO_TLE_ERR	6
ERR	Could not close an ID	Computation not performed	XO_CFI_GEN_TLE_CLOSE	7

			_ERR	
ERR	Could not write output file to disk	Computation not performed	XO_CFI_GEN_TLE_WRITE_FILE_ERR	8

7.46.6 Runtime performances

The following runtime performance has been measured.

Table 159: Runtime performances of xo_gen_tle function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
TBD	TBD	TBD	TBD

7.47 xo_check_osf

7.47.1 Overview

The **xo_check_osf** CFI function checks the continuity of the orbital parameters at the transition from one orbital change and the next one in an Orbit Scenario file.

7.47.2 Calling interface

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

```
#include <explorer_orbit.h>
{
       long    sat_id;
       xl_model_id model_id = {NULL};
       xl_time_id time_id = {NULL};
       char   *osf_file;
       long transition_number;
      double threshold[XO_NUM_CHECK_PARAMS],
             diffs[XO_NUM_CHECK_PARAMS];
       long status, ierr[XO_ERR_VECTOR_MAX_LENGTH];

       status = xo_check_osf(&sat_id, &model_id, &time_id,
                             osf_file, &transition_number,
                             threshold,
                             /* output */
                             diffs, ierr);

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

       status = xo_check_osf_run(&run_id,
                                 osf_file, &transition_number,
                                 threshold,
                                 /* output */
                                 diffs, ierr);
}
```

7.47.3 Input parameters

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

Table 160: Input parameters of xo_check_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
osf_file	char*	-	Orbit Scenario file to be checked	-	-
transition_ number	long*	-	Number of the transition to be checked. If 0, the last transition is checked	-	>1 <=Number of transitions
threshold	double [XO_NUM_CHECK_PARAMS]	0	Threshold for the time at ANX	s	>0
		1	Threshold for the ANX longitude	deg	
		2	Threshold for the MLST	s	
		3	Threshold for the osculating semi-axis major	m	
		4	Threshold for the osculating inclination	deg	
		5	Threshold for the nodal period	s	

7.47.4 Output parameters

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

Table 161: Output parameters of xo_check_osf function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_check_osf	long	-	Status	-	-1, 0, 1
diffs	double [XO_NUM_CHECK_PARAMS]	0	Difference for the time at ANX	s	>0
		1	Difference for the ANX longitude	deg	

		2	Difference for the MLST	s	
		3	Difference for the osculating semi-axis major	m	
		4	Difference for the osculating inclination	deg	
		5	Difference for the nodal period	s	
ierr	long*	all	Status vector	-	

7.47.5 Warnings and errors

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

Table 162: Error messages of xo_ckeck_osf function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Time correlations are not initialized	Computation not performed	XO_CFI_CHECK_OSF_TIME_INIT_ERR	0
ERR	Error reading the Orbit Scenario file	Computation not performed	XO_CFI_CHECK_OSF_OSF_READ_ERR	1
ERR	Wrong transition number	Computation not performed	XO_CFI_CHECK_OSF_WRONG_TRANSITION_ERR	2
ERR	Couldn't initialize the orbit	Computation not performed	XO_CFI_CHECK_OSF_ORBIT_INIT_ERR	3
ERR	Error in xo_orbit_info	Computation not performed	XO_CFI_CHECK_OSF_ORBIT_INFO_ERR	4
WARN	UTC at ANX exceeds the input threshold	Computation performed	XO_CFI_CHECK_OSF_UTC_WARN	5
WARN	ANX Longitude exceeds the input threshold	Computation performed	XO_CFI_CHECK_OSF_ANX_LONG_WARN	6
WARN	MLST exceeds the input threshold	Computation performed	XO_CFI_CHECK_OSF_MLST_WARN	7
WARN	Osculating semi-major axis exceeds the input threshold	Computation performed	XO_CFI_CHECK_OSF_OSC_A_WARN	8
WARN	Osculating inclination	Computation performed	XO_CFI_CHECK_OSF_OS	9

	exceeds the input threshold		C_I_WARN	
WARN	Nodal period exceeds the input threshold	Computation performed	XO_CFI_CHECK_OSF_TNOD_WARN	10

7.47.6 Runtime performances

The following runtime performance has been measured.

Table 163: Runtime performances of xo_check_osf function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
5.66	2.12	3.45	0.73

7.48 xo_check_oef

7.48.1 Overview

The **xo_check_oef** CFI function checks the consistency between the list of orbital changes and the list of state vectors in an Orbit Event file.

7.48.2 Calling interface

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

```
#include <explorer_orbit.h>
{
        long    sat_id;
        xl_time_id time_id = {NULL};
        xl_model_id model_id = {NULL};
        char   *oef_file;
        long    time_mode, time_ref;
        double start_time, stop_time;
        long     start_orbit, stop_orbit;
        double threshold[XO_NUM_CHECK_PARAMS],
               max_diffs[XO_NUM_CHECK_PARAMS],
               rms[XO_NUM_CHECK_PARAMS];
        long status, ierr[XO_ERR_VECTOR_MAX_LENGTH];
        status = xo_check_oef(&sat_id, &model_id, &time_id,
                              &time_mode, &time_ref,
                              &start_time, &stop_time,
                              &start_orbit, &stop_orbit,
                               oef_file, threshold,
                               /* output */
                               max_diffs, rms, ierr);

        /* Or, using the run_id */
        long run_id;
        status = xo_check_oef_run(&run_id,
                                  time_mode, &time_ref,
                                   &start_time, &stop_time,
                                   &start_orbit, &stop_orbit,
                                  oef_file, threshold,
                                  /* output */
                                  max_diffs, rms, ierr);
}
```

7.48.3 Input parameters

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

Table 164: Input parameters of xo_check_oef function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
sat_id	long *	-	Satellite ID	-	Complete
model_id	xl_model_id*	-	Model ID	-	-
time_id	xl_time_id*	-	Structure that contains the time correlations.	-	-
time_mode	long*	-	Flag for the input time range selection: whole file, time or orbits	-	XO_SEL_FILE, XO_SEL_TIME, XO_SEL_ORBIT
time_ref	long*	-	Time reference for the star_time and stop_time input parameters (only needed if time_mode is XO_SEL_TIME)	-	Complete
start_time	double*	-	Start time for the time range to be checked (only needed if time_mode is XO_SEL_TIME)	days	[-18262.0, 36524.0]
stop_time	double*	-	Stop time for the time range to be checked (only needed if time_mode is XO_SEL_TIME)	days	[-18262.0, 36524.0]
start_orbit	long*	-	Start orbit for the orbit range to be checked (only needed if time_mode is XO_SEL_ORBIT)	-	file range
stop_orbit	long*	-	Stop orbit for the orbit range to be checked (only needed if time_mode is XO_SEL_ORBIT)	-	file range
oef_file	char*	-	Orbit Event file to be checked	-	-
threshold	double [XO_NUM_CHECK_PARAMS]	0	Threshold for the time at ANX	s	>0
		1	Threshold for the ANX longitude	deg	
		2	Threshold for the MLST	s	
		3	Threshold for the osculating semi-axis major	m	
		4	Threshold for the osculating inclination	deg	
		5	Threshold for the nodal period	s	

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

- Satellite ID: sat_id.

- Time inputs selection: time_mode
- Time reference: time_ref.

7.48.4 Output parameters

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

Table 165: Output parameters of xo_check_oef function

C name	C type	Array Element	Description (Reference)	Unit (Format)	Allowed Range
xo_check_oef	long	-	Status	_	-1, 0, 1
max_diffs	double [XO_NUM_CHECK _PARAMS]	All	The following parameters are computed using the list of orbital changes and the list of state vectors. The maximum value of these differences for the requested interval are returned in this array.	_	>0
		0	Time at ANX	s	
		1	ANX longitude	deg	
		2	MLST	s	
		3	Osculating semi-axis major	m	
		4	Osculating inclination	deg	
		5	Nodal period	s	
rms	double [XO_NUM_CHECK _PARAMS]	All	The following parameters are computed using the list of orbital changes and the list of state vectors. The standard deviation of these differences for the requested interval are returned in this array.	_	>0
		1	ANX longitude	deg	
		2	MLST	s	
		3	Osculating semi-axis major	m	
		4	Osculating inclination	deg	
		5	Nodal period	s	
ierr	long*	all	Status vector	_	

7.48.5 Warnings and errors

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

Table 166: Error messages of xo_ckeck_oef function

Error type	Error message	Cause and impact	Error code	Error No
ERR	Error in xo_orbit_info	Computation not performed	XO_CFI_CHECK_OEF_ORBIT_INIT_ERR	0
ERR	Couldn't initialize the orbit	Computation not performed	XO_CFI_CHECK_OEF_ORBIT_INFO_ERR	1
ERR	Memory allocation error	Computation not performed	XO_CFI_CHECK_OEF_MEM_ERR	2
WARN	UTC at ANX exceeds the input threshold	Computation performed	XO_CFI_CHECK_OEF_UTC_WARN	3
WARN	ANX Longitude exceeds the input threshold	Computation performed	XO_CFI_CHECK_OEF_ANX_LONG_WARN	4
WARN	MLST exceeds the input threshold	Computation performed	XO_CHECK_OEF_MLST_W ARN	5
WARN	Osculating semi-major axis exceeds the input threshold	Computation performed	XO_CFI_CHECK_OEF_OSC_A_WARN	6
WARN	Osculating inclination exceeds the input threshold	Computation performed	XO_CFI_CHECK_OEF_OSC_I_WARN	7
WARN	Nodal period exceeds the input threshold	Computation performed	XO_CFI_CHECK_OEF_TNOD_WARN	8

7.48.6 Runtime performances

The following runtime performance has been measured.

Table 167: Runtime performances of xo_check_oef function

Solaris 32-bit. [ms]	Solaris 64 bit. [ms]	Linux 32-bit. [ms]	Linux 64-bit. [ms]
62.6	20.6	26.6	6.0

8 LIBRARY PRECAUTIONS

The following precautions shall be taken into account when using EO_ORBIT software library:

- When a message like

  <LIBRARY NAME> >>> ERROR in *xo_function*: Internal computation error # *n*

  or

  <LIBRARY NAME> >>> WARNING in *xo_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 168: Known problems list

CFI library	Problem	Work around solution
Spot model for propagation	Functionality is not currently available	-
xo_orbit_get_osv_compute_validity	The validity interval returned by the function is not correct	The correct validity interval is defined in section 7.28.2