NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY

NGA.STND.0028-2_1.0
2012-08-22

Lidar Data and Metadata: Conceptual Model and Metadata Dictionary for Enterprise Level 2 Volume Return Products

Version 1.0

2012-08-22

NATIONAL CENTER FOR GEOSPATIAL INTELLIGENCE STANDARDS

Revision Summary

Title	Date	Description
lidar Metadata: Conceptual Model and Metadata Dictionary 0.1	Feb2009	First Draft of lidar Conceptual model. Uses NMF format and LAS content plus organizational contact/role data
lidar Data and Metadata: Conceptual Model and Metadata Dictionary 0.2W	Jun2009	Working copy of second Draft. Title extended to cover wideband as well as metadata documentation. Substantial additions to over view discussion in section 8. Introduces "Levels" as well as stages (new name for previous lidar "level" nomenclature) to describe lidar products. Adds a 27 page physical lidar model in section 10, fills in missing technical content from LAS 1.x to 2.0 documentation including a general scan parameterization, waveform clips, tiling, transfer of pulse level data from point records to pulse records (a major data volume reducer), hierarchical tag naming, elimination of non- lidar data below level-4 products,…. 85 New metadata items and mapping to old data is imbedded in a spread sheet at top of section 9. Provisional tile mapping data.
lidar Data and Metadata: Conceptual Model and Metadata Dictionary 0.2	Jul2009	Embedded spread sheet migrated to existing data tables. Domain and Essential/not table entries remain largely unfilled. Detailed definition paragraph stubs and hyperlinks for each brief to detailed definition stub inserted using prototype of MD.Meta.DS_Name. Make various updates and supply open issue closure on tile map. Table of Contents for Appendix 20 now displays as the data tree.
lidar Data and Metadata: Conceptual Model and Metadata Dictionary 0.3W	AUG2009	Working Copy of 3rd draft has Substantial population of detailed definition tables. "Domain" and "Essential/not"populated for every tag with a detailed definition. Deletion of grayed tables from version 0.1.
lidar Data and Metadata: Conceptual Model and Metadata Dictionary 0.31W	NOV2009	Detailed tables provisionally filled in for part of MD.*. All entries provided for ES.*, SC.*, LD.*
lidar Data and Metadata: Conceptual Model and Metadata Dictionary 0.39	DEC2009	More complete detailed definitions and DAL access documentation for standardized reading and writing if lidar files. Option to break into separate level-1, level-2, level-3, level-4 class based documents
CMMD0.4	APR2010	Last Version before level break out. Added User defined data, channelization stubbs. Channelization will be worked more extensively as L-1 and L-2 breakout issues.

CMMD L2 0.41	MAY2010	Begin as copy of CMMD0.4. 1)Mark non-L-2 items as “formerly” 2)Review paragraphs and make final deletion of non-L2 data. 3) modify section 9 and 10 to match deletions in section 20.
CMMD L2 0.65	MAR2011	Added sensor based pts (angle angle range) Deleted non-L2 content paragraphs.
CMMD L2 0.9	MAR2011	Readied for GWG. Reduced pulse indexing for waveforms (“ipi”) transitioned. Consistency checked
CMMD L2 0.9	AUG2011	Adopted changes due to L1 comments from L1 vetting by NTB
CMMD L2 0.9	SEP2011	Temporal Dynamics discovery metadata incorporated.
CMMD L2. 1.0	NOV2011	Added Synthetic Aperture complex data grid after LD.Point.*. Also added slots for ES.SAL* and PF.SAL.*
CMMD L2 1.0	AUG2012	Additional detector array properties documentation, enhanced error decorrelation model, enhanced geometric adjustment model, support for decomposition of range error based on physical source, added factors to convert std. deviations to percentiles, upgraded to full stokes vector support for polarization channelization,

Table of Contents

1. OVERVIEW 1

1.1. EXECUTIVE SUMMARY OF ENTERPRISE LEVEL-2 LIDAR MODEL 1

2. SCOPE 5

3. CONFORMANCE 8

3.1. EXTERNAL RELATED DATA STANDARDS DOCUMENTS 8
3.2. REGISTRY RECOMENDATIONS 8

4. REFERENCES 9

4.1. NORMATIVE REFERENCES 9
4.2. INFORMATIVE REFERENCES 10

5. TERMS, DEFINITIONS AND CONVENTIONS 10

5.1. TERMS AND DEFINITIONS 10
5.2. CONVENTIONS 20

6. DATA TYPES 23

6.1. ABSTRACT DATA TYPES 23

7. CONCEPTUAL MODEL LAYOUT 26

7.1. CONCEPTUAL MODEL STRUCTURE DIAGRAM 26
7.2. CONCEPTUAL MODEL METADATA SECTION TABLES 27

8. LIDAR CONCEPTUAL MODEL OVERVIEW 29

8.1. OVERVIEW OF THE PHYSICAL MODEL OF LIDAR VRP CLASS 29
8.1.1 Sub-models 29
8.2. OVERVIEW OF THE LIDAR CONCEPTUAL MODEL DATA RELATIONS. 32
8.2.1 Top Level (Class) Data Categories 32

9. DATA STRUCTURE OF THE LIDAR CONCEPTUAL MODEL 34

9.1. MD.* TRUE METADATA 34
9.2. ES –EXPLOITATION SUPPORT 45
9.3. PF –PRODUCT FORMATION 64
9.4. PD –PRODUCT DEFINITION 74
9.5. SC –SENSOR CONFIGURATION 90
9.6. LD –LIDAR DATA 102
9.7. UD –USER DEFINED WIDE BAND DATA 103
9.8. CREATING PROFILES OF THE CMMD 104

10. APPENDIX: LIDAR PHYSICAL MODEL DISCUSSION 108

10.1. SENSOR PHYSICS AND PHYSICAL OPERATION 108
10.1.1 Timing Model 108
10.1.2 Beam Model 120
10.1.3 Channelization Model 129
10.1.4 Sensor Measurements, Hierarchical Organization (PCD, Scan, Subswath and Swath) 137
10.1.5 Data Support for Geolocation error Propagation 154
10.2. LIDAR DATA STRATEGIES 168
10.2.1 Lidar Physical Data Blocking: Tiles 168
10.3. GENERALIZATION OF VRP DATA 190
10.3.1 Fast time data—Waveform capture 190
10.4. USER DEFINED DATA 193

20. APPENDIX: LIDAR DATA DICTIONARY-- DETAILED DATA ITEM DEFINITIONS ... 195

20.1 MD.* TRUE METADATA ... 195
MD.Aggregate.* ... 195
20.1.1 MD.Aggregate.Parent.CollctnDate(j) [L2,3,4,5] ... 195
20.1.2 MD.Aggregate.Parent.CollctnOpID(j) [L2,3,4,5] ... 197
20.1.3 MD.Aggregate.Parent.DS_Name(j) [L2,3,4,5] ... 198
20.1.4 MD.Aggregate.Parent.Duration(j) [L2,3,4,5] ... 199
20.1.5 MD.Aggregate.Parent.StartTime(j) [L2,3,4,5] ... 199
20.1.6 MD.Aggregate.Fellow.CollctnDate(j) [L1,2,3] ... 201
20.1.7 MD.Aggregate.Fellow.CollctnOpID(j) [L1,2,3,4,5] ... 202
20.1.8 MD.Aggregate.Fellow.DS_Name(j) [L1,2,3,4,5] ... 203
20.1.9 MD.Aggregate.Fellow.PlatformID(j) [L1,2,3,4,5] ... 203
20.1.10 MD.Aggregate.Fellow.ProdType(j) [L1,2,3,4,5] ... 204
20.1.11 MD.Aggregate.Fellow.SensorID(j) [L1,2,3,4,5] ... 204
20.1.12 MD.Aggregate.Fellow.SensorType(j) [L1,2,3,4,5] ... 205
20.1.13 MD.Aggregate.Fellow.StartTime(j) [L1,2,3,4,5] ... 206
MD.Collect.* ... 207
20.1.14 MD.Collect.Duration [L1,2,3,5] ... 207
MD.Coverage.* ... 207
20.1.15 MD.Coverage.BE_NumList(j) [L1,2,3,4,5] ... 208
20.1.16 MD.Coverage.Bounds.LatNorth [L1,2,3,4,5] ... 208
20.1.17 MD.Coverage.Bounds.LatSouth [L1,2,3,4,5] ... 208
20.1.18 MD.Coverage.Bounds.LonEast [L1,2,3,4,5] ... 209
20.1.19 MD.Coverage.Bounds.LonWest[L1,2,3,4,5] ... 209
20.1.20 MD.Coverage.Bounds.ZoneList(j) [L1,2,3,4,5] ... 209
20.1.21 MD.Coverage.Bounds.OuterZoneList(j) [L1,2,3,4,5] ... 211
20.1.22 MD.Coverage.Bounds.ZoneRanges(j) [L1,2,3,4,5] ... 212
20.1.23 MD.Coverage.Bounds.Vertices(iv,j) [L1,2,3,4,5] ... 213
20.1.24 MD.Coverage.Bounds.Z(j) [L1,2,3,4,5] ... 214
20.1.25 MD.Coverage.Centroid[L1,2,3,4,5] ... 215
20.1.26 MD.Coverage.Cntry.Cd2(j) [L1,2,3,4,5] ... 215
20.1.27 MD.Coverage.Cntry.Cd3(j) [L1,2,3,4,5] ... 216
20.1.28 MD.Coverage.Temporal.Duration [L2,3,4,5] ... 216
20.1.29 MD.Coverage.Temporal.StartDate [L2,3,4,5] ... 217
20.1.30 MD.Coverage.Temporal.StartTime [L2,3,4,5] ... 219
20.1.31 MD.Coverage.Voids.Atmsphrc(j,k) [L1,2,3] ... 219
20.1.32 MD.Coverage.Voids.Incompl(j,k) [L1,2,3] ... 220
20.1.33 MD.Coverage.Voids.Masked(j,k) [L1,2,3] ... 221
20.1.34 MD.Coverage.Voids.UnColl(j,k) [L1,2,3] ... 223
MD.Dynamics.* ... 224
20.1.35 MD.Dynamics.Exposure [L2,3,4,5] ... 224
20.1.36 MD.Dynamics.FrameRate [L2,3,4,5] ... 224
20.1.37 MD.Dynamics.LS_PanCode(j) [L2,3,4,5] ... 225
20.1.38 MD.Dynamics.LS_PanRate(j) [L2,3,4,5] ... 226
20.1.39 MD.Dynamics.LS_PatchDat(j) [L2,3,4,5] ... 227
20.1.40 MD.Dynamics.LS_PatchDwell(j) [L2,3,4,5] ... 229
20.1.41 MD.Dynamics.VisitsPerPatch(j) [L2,3,4,5] ... 229
MD.File.* ... 230
20.1.42 MD.File.Class[L1,2,3,4,5] ... 231
20.1.43 MD.File.ClassBy[L1,2,3,4,5] ... 231
20.1.44 MD.File.Comp[L1,2,3,4,5] ... 232
20.1.45 MD.File.DeclDate[L1,2,3,4,5] ... 232
20.1.46 MD.File.DeclExemtn[L1,2,3,4,5] ... 233
20.1.47 MD.File.DeclMeth[L1,2,3,4,5] ... 233
20.1.48 MD.File.DissContrls(j) [L1,2,3,4,5] ... 233
20.1.49 MD.File.DG[L1,2,3,4,5] ... 234
20.1.50 MD.File.DG_Date[L1,2,3,4,5] ... 234
20.1.51 MD.File.Reason[L1,2,3,4,5] ... 235
MD.InfAssur.* ... 235
20.1.52 Candidate MD.InfAssur.Seal [L1,2,3,4,5] ... 235

20.1.53 Candidate MD.InfAssur.NonRepud [L1,2,3,4,5] 236
MD.LegalConstr.* 236
20.1.54 MD.LegalConstr.AttributionCitation[L1,2,3,4,5] 236
20.1.55 MD.LegalConstr.Derivation(j) [L1,2,3,4,5] 236
20.1.56 MD.LegalConstr.Dissem(j) [L1,2,3,4,5] 237
MD.Meta.* 237
20.1.57 MD.Meta.Class[L1,2,3,4,5] 237
20.1.58 MD.Meta.DAL_VersionName(j) [L1,2,3,4,5] 238
20.1.59 MD.Meta.DR_ContactAddr[L1,2,3,4,5] 239
20.1.60 MD.Meta.DS_Name [L1,2,3,4,5] 240
20.1.61 MD.Meta.ID[L1,2,3,4,5] 242
20.1.62 MD.Meta.Language[L1,2,3,4,5] 243
20.1.63 MD.Meta.MaintOrg[L1,2,3,4,5] 243
MD.NomGeo.* 243
20.1.64 MD.NomGeo.Pt.Az.Ctr [L1,2,3,4,5] 243
20.1.65 MD.NomGeo.Pt.Az.Spread [L1,2,3,4,5] 245
20.1.66 MD.NomGeo.Pt.Nadir.Ctr [L1,2,3,4,5] 246
20.1.67 MD.NomGeo.Pt.Nadir.Spread [L1,2,3,4,5] 247
20.1.68 MD.NomGeo.Swath.Az.Ctr [L1,2,3,4,5] 248
20.1.69 MD.NomGeo.Swath.Az.Spread [L1,2,3,4,5] 249
20.1.70 MD.NomGeo.Swath.Nadir.Ctr [L1,2,3,4,5] 250
20.1.71 MD.NomGeo.Swath.Nadir.Spread [L1,2,3,4,5] 252
MD.Objective.* 253
20.1.72 MD.Objective.Abstract [L1,2,3,4,5] 253
20.1.73 MD.Objective.BE_NumList(j) [L1,2,3,4,5] 254
20.1.74 MD.Objective.Conops(j) [L1,2,3,4,5] 254
20.1.75 MD.Objective.DateDue[L1,2,3,4,5] 256
20.1.76 MD.Objective.DateAcceptable[L1,2,3,4,5] 257
20.1.77 MD.Objective.Desc[L1,2,3,4,5] 258
20.1.78 MD.Objective.EastBound[L1,2,3,4,5] 258
20.1.79 MD.Objective.ID [L1,2,3,4,5] 259
20.1.80 MD.Objective.NorthBound [L1,2,3,4,5] 259
20.1.81 MD.Objective.SouthBound [L1,2,3,4,5] 259
20.1.82 MD.Objective.Tasker[L1,2,3,4,5] 260
20.1.83 MD.Objective.Vertices(iv,j) [L1,2,3,4,5] 260
20.1.84 MD.Objective.WAC_List(j,k) [L1,2,3,4,5] 261
20.1.85 MD.Objective.WBStyle [L1,2,3,4,5] 262
20.1.86 MD.Objective.WestBound [L1,2,3,4,5] 263
MD.Platform.* 264
20.1.87 MD.Platform.Desc(j) [L1,2,3,4,5] 264
20.1.88 MD.Platform.ID(j) [L1,2,3,4,5] 264
MD.ProcHist.* 265
20.1.89 MD.ProcHist.CodeList(j) [L1,2,3,4,5] 265
20.1.90 MD.ProcHist.L1Writer.Name(k) [L1,2,3,4] 266
20.1.91 MD.ProcHist.L1Writer.ProcDate(k) [L1,2,3,4] 266
20.1.92 MD.ProcHist.L1Writer.ProcSite(k) [L1,2,3,4] 267
20.1.93 MD.ProcHist.L1Writer.RelDate(k) [L1,2,3,4] 268
20.1.94 MD.ProcHist.L1Writer.Vendor(k) [L1,2,3] 269
20.1.95 MD.ProcHist.L1Writer.Versn(k) [L1,2,3,4] 269
20.1.96 MD.ProcHist.L2App.Name(j,k) [L2,3,4] 270
20.1.97 MD.ProcHist.L2App.ProcDate(j,k) [L2,3,4] 271
20.1.98 MD.ProcHist.L2App.ProcSite(j,k) [L2,3,4] 272
20.1.99 MD.ProcHist.L2App.RelDate(j,k) [L2,3] 272
20.1.100 MD.ProcHist.L2App.Vendor(j,k) [L2,3,4] 274
20.1.101 MD.ProcHist.L2App.Versn(j,k) [L2,3,4] 275
20.1.102 MD.ProcHist.L2NumStages(k)[L2,3,4] 275
20.1.103 MD.ProcHist.ProdType(j) [L1,2,3] 277
MD.Qual.* 279
20.1.104 MD.Qual.GeoLocUncH [L1,2,3,4,5] 279
20.1.105 MD.Qual.GeoLocUncV[L1,2,3,4,5] 280
20.1.106 MD.Qual.GSD [L1,2,3,4,5] 281

20.1.107 MD.Qual.RetPtDensity[L1,2,3,4,5] 282
20.1.108 MD.Qual.Impairments(j) [L2,3,4,5] 283
20.1.109 MD.Qual.XrangeAvDia(j) [L2,3,4,5] 285
MD.Range.* 286
20.1.110 MD.Range.Max[L1,2,3,4,5] 286
20.1.111 MD.Range.Min[L1,2,3,4,5] 286
20.1.112 MD.Range.NomRgToScene [L1,2,3,4,5] 287
MD.Sensor.* 287
20.1.113 MD.Sensor.ID(j) [L1,2,3,4,5] 287
20.1.114 MD.Sensor.LidarMode(j) [L1,2,3,] 288
20.1.115 MD.Sensor.lidarType(j) [L1,2,3] 289
MD.Truth.* 290
20.1.116 MD.Truth.Dat(j) [L1,2,3,4,5] 290
20.1.117 MD.Truth.Descr [L1,2,3,4,5] 292
MD.UseHist.* 293
20.1.118 MD.UseHist.Date(j) [L1,2,3,4,5] 293
20.1.119 MD.UseHist.Agent(j) [L1,2,3,4,5] 294
20.1.120 MD.UseHist.Purpose(j) [L1,2,3,4,5] 295
20.1.121 MD.UseHist.Status(j) [L1,2,3,4,5] 295
20.2 ES.* EXPLOITATION SUPPORT 296
ES.Align.* 296
20.2.1 ES.Align.LocG_Vert(j_tile, j) [L2,3,4,5] 296
20.2.2 ES.Align.X_ECEF(j_tile, j) [L2,3,4,5] 297
20.2.3 ES.Align.Y_ECEF(j_tile,j) [L2,3,4,5] 297
20.2.4 ES.Align.Z_ECEF(j_tile,j) [L2,3,4,5] 298
20.2.5 ES.Align.XYZ_Flag [L2,3,4,5] 299
ES.CollCondn.* 300
20.2.6 ES.CollCondn.Obscurant [L1,2,3] 300
20.2.7 ES.CollCondn.PercentAreaAffected [L1, 2,3] 301
ES.Offset.* 301
20.2.8 ES.Offset.X(j_tile) [L2,3,4,5] 302
20.2.9 ES.Offset.Y(j_tile) [L2,3,4,5] 302
20.2.10 ES.Offset.Z(j_tile) [L2,3,4,5] 303
ES.Origin.* 304
20.2.11 ES.Origin.X_ECEF(j_tile) [L2,3,4,5] 304
20.2.12 ES.Origin.Y_ECEF(j_tile) [L2,3,4,5] 305
20.2.13 ES.Origin.Z_ECEF(j_tile) [L2,3,4,5] 306
20.2.14 ES.Origin.GeodtcEpoch(j) [L1,2,3,4,5] 306
20.2.15 ES.Origin.X_MGRS(j_tile) [L2,3,4,5] 307
20.2.16 ES.Origin.Y_MGRS(j_tile) [L2,3,4,5] 308
20.2.17 ES.Origin.Z_MGRS(j_tile) [L2,3,4,5] 309
20.2.18 ES.Origin.MGRS_Flag[L2,3,4,5] 310
ES.PDOP.* 311
20.2.19 ES.PDOP.Max[L2,3,4,5] 312
20.2.20 ES.PDOP.Min[L2,3,4,5] 312
20.2.21 ES.PDOP.Val(j_tile) [L2,3,4,5] 312
ES.Point.* 313
20.2.22 ES.Point.CharCode.*TileNum*(ipt) [L2,3,4,5] 313
20.2.23 ES.Point.FilterScore.*TileNum*(ipt) [L2,3] 314
20.2.24 ES.Point.IntensyFlg [L2,3,4,5] 315
20.2.25 ES.Point.IntnsyRemap(j,k) [L2,3,4,5] 316
20.2.26 ES.Point.LookSpread.*TileNum* [L2,3,4,5] 318
20.2.27 ES.Point.LM_SigmaR.*TileNum*(ipt) [L2,3,4,5] 319
20.2.28 ES.Point.LM_SigmaT.*TileNum*(ipt) [L2,3,4,5] 319
20.2.29 ES.Point.PulseNum.*TileNum*(ipt, j) [L2,3,4,5] 320
20.2.30 ES.Point.RtnsCode.*TileNum*(ipt, j) [L2,3,4,5] 322
20.2.31 ES.Point.SigmaXYZ(j,k) [L2,3,4,5] 324
20.2.32 ES.Point.TruthTags.*Tilenum*(ipt,j) [L2,3,4,5] 326
ES.SAL.* 328
20.2.33 ES.SAL.GridAxes(j) [L2] 328
20.2.34 ES.SAL.GridType [L2] 328

20.2.35 ES.SAL.IPR(j) [L2] 329
20.2.36 ES.SAL.ORP(j) [L2] 329
20.2.37 ES.SAL.SL_RgAxis(j) [L2] 329
20.2.38 ES.SAL.SL_AzAxis(j) [L2] 329
20.2.39 ES.SAL.StepSize(j) [L2] 330
ES.Scale.* 330
20.2.40 ES.Scale.X_Fac [L2,3,4,5] 330
20.2.41 ES.Scale.Y_Fac[L2,3,4,5] 331
20.2.42 ES.Scale.Z_Fac [L2,3,4,5] 332
ES.Scan.* 333
20.2.43 ES.Scan.CorrParms.Alpha(j,k) [L1,2,3,4,5] 333
20.2.44 ES.Scan.CorrParms.Beta(j,k) [L1,2,3,4,5] 335
20.2.45 ES.Scan.CorrParms.Gamma(j,k) [L1,2,3,4,5] 338
20.2.46 ES.Scan.CorrParms.X(j,k) [L1,2,3,4,5] 341
20.2.47 ES.Scan.CorrParms.Y(j,k) [L1,2,3,4,5] 344
20.2.48 ES.Scan.CorrParms.Z(j,k) [L1,2,3,4,5] 347
20.2.49 ES.Scan.CorrParms.R(j,n,k) [L1,2,3,4,5] 350
20.2.50 ES.Scan.CorrParms.AlphaBeta(j,k) [L1,2,3,4,5] 353
20.2.51 ES.Scan.CorrParms.AlphaGamma(j,k) [L1,2,3,4,5] 356
20.2.52 ES.Scan.CorrParms.BetaGamma(j,k) [L1,2,3,4,5] 358
20.2.53 ES.Scan.CorrParms.XY(j,k) [L1,2,3,4,5] 359
20.2.54 ES.Scan.CorrParms.XZ(j,k) [L1,2,3,4,5] 361
20.2.55 ES.Scan.CorrParms.XAlpha(j,k) [L1,2,3,4,5] 362
20.2.56 ES.Scan.CorrParms.XBeta(j,k) [L1,2,3,4,5] 364
20.2.57 ES.Scan.CorrParms.XGamma(j,k) [L1,2,3,4,5] 365
20.2.58 ES.Scan.CorrParms.YZ(j,k) [L1,2,3,4,5] 367
20.2.59 ES.Scan.CorrParms.YAlpha(j,k) [L1,2,3,4,5] 368
20.2.60 ES.Scan.CorrParms.YBeta(j,k) [L1,2,3,4,5] 370
20.2.61 ES.Scan.CorrParms.YGamma(j,k) [L1,2,3,4,5] 372
20.2.62 ES.Scan.CorrParms.ZAlpha(j,k) [L1,2,3,4,5] 373
20.2.63 ES.Scan.CorrParms.ZBeta(j,k) [L1,2,3,4,5] 375
20.2.64 ES.Scan.CorrParms.ZGamma(j,k) [L1,2,3,4,5] 376
20.2.65 ES.Scan.CorrParmsT.Alpha(j,k) [L1,2,3,4,5] 378
20.2.66 ES.Scan.CorrParmsT.Beta(j,k) [L1,2,3,4,5] 381
20.2.67 ES.Scan.CorrParmsT.Gamma(j,k) [L1,2,3,4,5] 383
20.2.68 ES.Scan.CorrParmsT.X(j,k) [L1,2,3,4,5] 386
20.2.69 ES.Scan.CorrParmsT.Y(j,k) [L1,2,3,4,5] 388
20.2.70 ES.Scan.CorrParmsT.Z(j,k) [L1,2,3,4,5] 391
20.2.71 ES.Scan.CorrParmsT.R(j,k) [L1,2,3,4,5] 394
20.2.72 ES.Scan.CorrParmsT.AlphaBeta(j,k) [L1,2,3,4,5] 396
20.2.73 ES.Scan.CorrParmsT.AlphaGamma(j,k) [L1,2,3,4,5] 399
20.2.74 ES.Scan.CorrParmsT.BetaGamma(j,k) [L1,2,3,4,5] 402
20.2.75 ES.Scan.CorrParmsT.XY(j,k) [L1,2,3,4,5] 404
20.2.76 ES.Scan.CorrParmsT.XZ(j,k) [L1,2,3,4,5] 407
20.2.77 ES.Scan.CorrParmsT.XAlpha(j,k) [L1,2,3,4,5] 410
20.2.78 ES.Scan.CorrParmsT.XBeta(j,k) [L1,2,3,4,5] 412
20.2.79 ES.Scan.CorrParmsT.XGamma(j,k) [L1,2,3,4,5] 415
20.2.80 ES.Scan.CorrParmsT.YZ(j,k) [L1,2,3,4,5] 417
20.2.81 ES.Scan.CorrParmsT.YAlpha(j,k) [L1,2,3,4,5] 420
20.2.82 ES.Scan.CorrParmsT.YBeta(j,k) [L1,2,3,4,5] 423
20.2.83 ES.Scan.CorrParmsT.YGamma(j,k) [L1,2,3,4,5] 425
20.2.84 ES.Scan.CorrParmsT.ZAlpha(j,k) [L1,2,3,4,5] 428
20.2.85 ES.Scan.CorrParmsT.ZBeta(j,k) [L1,2,3,4,5] 431
20.2.86 ES.Scan.CorrParmsT.ZGamma(j,k) [L1,2,3,4,5] 433
20.2.87 ES.Scan.DelT_Cov [L1,2,3,4,5] 436
20.2.88 ES.Scan.GM_SigmaR(j) [L1,2,3,4,5] 437
20.2.89 ES.Scan.RhoAlphaGamma(j) [L1,2,3,4,5] 439
20.2.90 ES.Scan.RhoXBeta(j) [L1,2,3,4,5] 440
20.2.91 ES.Scan.RhoAlphaBeta(j) [L1,2,3,4,5] 441
20.2.92 ES.Scan.RhoBetaGamma(j) [L1,2,3,4,5] 442
20.2.93 ES.Scan.RhoXAlpha(j) [L1,2,3,4,5] 443
20.2.94 ES.Scan.RhoXGamma(j) [L1,2,3,4,5] 444

20.2.95 ES.Scan.RhoXY(j) [L1,2,3,4,5] 446
20.2.96 ES.Scan.RhoXZ(j) [L1,2,3,4,5] 447
20.2.97 ES.Scan.RhoYAlpha(j) [L1,2,3,4,5] 448
20.2.98 ES.Scan.RhoYBeta(j) [L1,2,3,4,5] 449
20.2.99 ES.Scan.RhoYGamma(j) [L1,2,3,4,5] 450
20.2.100 ES.Scan.RhoYZ(j) [L1,2,3,4,5] 451
20.2.101 ES.Scan.RhoZAlpha(j) [L1,2,3,4,5] 452
20.2.102 ES.Scan.RhoZBeta(j) [L1,2,3,4,5] 453
20.2.103 ES.Scan.RhoZGamma(j) [L1,2,3,4,5] 454
20.2.104 ES.Scan.SigmaAlpha(j) [L1,2,3,4,5] 455
20.2.105 ES.Scan.SigmaBeta(j) [L1,2,3,4,5] 456
20.2.106 ES.Scan.SigmaGamma(j) [L1,2,3,4,5] 457
20.2.107 ES.Scan.SigmaX(j) [L1,2,3,4,5] 458
20.2.108 ES.Scan.SigmaY(j) [L1,2,3,4,5] 459
20.2.109 ES.Scan.SigmaZ(j) [L1,2,3,4,5] 461
20.2.110 ES.Scan.SigmaNbar(j) [L1,2,3,4,5] 462
ES.Swath.* 462
20.2.111 ES.Swath.CorrParms.Alpha(j,k) [L2,3,4,5] 462
20.2.112 ES.Swath.CorrParms.Beta(j,k) [L2,3,4,5] 465
20.2.113 ES.Swath.CorrParms.Gamma(j,k) [L2,3,4,5] 468
20.2.114 ES.Swath.CorrParms.X(j,k) [L2,3,4,5] 470
20.2.115 ES.Swath.CorrParms.Y(j,k) [L2,3,4,5] 474
20.2.116 ES.Swath.CorrParms.Z(j,k) [L2,3,4,5] 478
20.2.117 ES.Swath.CorrParms.R(j,n,k) [L2,3,4,5] 482
20.2.118 ES.Swath.CorrParms.AlphaBeta(j,k) [L2,3,4,5] 485
20.2.119 ES.Swath.CorrParms.AlphaGamma(j,k) [L2,3,4,5] 488
20.2.120 ES.Swath.CorrParms.BetaGamma(j,k) [L2,3,4,5] 489
20.2.121 ES.Swath.CorrParms.XY(j,k) [L2,3,4,5] 491
20.2.122 ES.Swath.CorrParms.XZ(j,k) [L2,3,4,5] 492
20.2.123 ES.Swath.CorrParms.XAlpha(j,k) [L2,3,4,5] 494
20.2.124 ES.Swath.CorrParms.XBeta(j,k) [L2,3,4,5] 495
20.2.125 ES.Swath.CorrParms.XGamma(j,k) [L2,3,4,5] 497
20.2.126 ES.Swath.CorrParms.YZ(j,k) [L2,3,4,5] 499
20.2.127 ES.Swath.CorrParms.YAlpha(j,k) [L2,3,4,5] 500
20.2.128 ES.Swath.CorrParms.YBeta(j,k) [L2,3,4,5] 502
20.2.129 ES.Swath.CorrParms.YGamma(j,k) [L2,3,4,5] 503
20.2.130 ES.Swath.CorrParms.ZAlpha(j,k) [L2,3,4,5] 505
20.2.131 ES.Swath.CorrParms.ZBeta(j,k) [L2,3,4,5] 506
20.2.132 ES.Swath.CorrParms.ZGamma(j,k) [L2,3,4,5] 508
20.2.133 ES.Swath.DelT_Cov[L2,3,4,5] 509
20.2.134 ES.Swath.GM_SigmaRfrac(i,j) [L2,3,4,5] 510
20.2.135 ES.Swath.GM_SigmaT [L1,2,3,4,5] 512
20.2.136 ES.Swath.RhoAlphaGamma(k) [L2,3,4,5] 514
20.2.137 ES.Swath.RhoXBeta(k) [L2,3,4,5] 515
20.2.138 ES.Swath.RhoAlphaBeta(k) [L2,3,4,5] 516
20.2.139 ES.Swath.RhoBetaGamma(k) [L2,3,4,5] 517
20.2.140 ES.Swath.RhoXAlpha(k) [L2,3,4,5] 518
20.2.141 ES.Swath.RhoXGamma(k) [L2,3,4,5] 520
20.2.142 ES.Swath.RhoXY(k) [L2,3,4,5] 521
20.2.143 ES.Swath.RhoXZ(k) [L2,3,4,5] 522
20.2.144 ES.Swath.RhoYAlpha(k) [L2,3,4,5] 523
20.2.145 ES.Swath.RhoYBeta(k) [L2,3,4,5] 524
20.2.146 ES.Swath.RhoYGamma(k) [L2,3,4,5] 525
20.2.147 ES.Swath.RhoYZ(k) [L2,3,4,5] 526
20.2.148 ES.Swath.RhoZAlpha(k) [L2,3,4,5] 527
20.2.149 ES.Swath.RhoZBeta(k) [L2,3,4,5] 528
20.2.150 ES.Swath.RhoZGamma(k) [L2,3,4,5] 529
20.2.151 ES.Swath.SigmaAlpha(k) [L2,3,4,5] 530
20.2.152 ES.Swath.SigmaBeta(k) [L2,3,4,5] 531
20.2.153 ES.Swath.SigmaGamma(k) [L2,3,4,5] 532
20.2.154 ES.Swath.SigmaX(k) [L2,3,4,5] 533
20.2.155 ES.Swath.SigmaY(k) [L2,3,4,5] 534

20.2.156 ES.Swath.SigmaZ(k) [L2,3,4,5] ... 535
20.2.157 ES.Swath.SensorBasedCoordFlg(j) [L2,3,4,5] ... 536
20.2.158 ES.Swath.HiConfdncP_Fac(i,j,k) [L1,2,3,4,5] ... 539
20.2.159 ES.Swath.PSF(j,k) [L1,2,3,4,5] ... 544
ES.Truth.* ... 546
20.2.160 ES.Truth.IllusKMZ.*j* [L1,2,3,4,5] ... 546
20.2.161 ES.Truth.CalChRatios(ich) [L2,3,4,5] ... 546
20.2.162 ES.Truth.IntnsyRef(itp,ich,j) [L1,2] ... 548
20.2.163 ES.Truth.Pos(itp,j) [L1,2,3,4,5] ... 550
20.2.164 ES.Truth.CovCorrs(itp, j) [L1,2,3,4,5] ... 551
20.2.165 ES.Truth.CovSigs(itp, j) [L1,2,3,4,5] ... 552
ES.WFClp.* ... 552
20.2.166 ES.WFClp.ChNum.*TileNum*(ipi,iray) [L1,2] ... 552
20.2.167 ES.WFClp.Locs.*TileNum*(ipi,iray, j) [L2] ... 555
20.2.168 ES.WFClp.RtnsCode.*TileNum*(iray/irays, ipi/j) [L2] ... 558
20.3 PF.* PRODUCT FORMATION ... 564
PF.Adjustmts.* ... 564
20.3.1 PF.Adjustmts.Artifacts(j) [L2,3,4,5] ... 565
20.3.2 PF.Adjustmts.Mean.Pulse.*TileNum*(ip, j) [L2,3,4,5] ... 565
20.3.3 PF.Adjustmts.Mean.Scan(is,j) [L2,3,4,5] ... 567
20.3.4 PF.Adjustmts.Mean.Subswath.*Iswl*(isw,j) [L2,3,4,5] ... 568
20.3.5 PF.Adjustmts.Diffs.Positions.Scan(is,i,j) [L2,3,4,5] ... 571
20.3.6 PF.Adjustmts.Diffs.Positions.SubSwath.*Iswl*(isw,i,j) [L2,3,4,5] ... 572
20.3.7 PF.Adjustmts.Diffs.Shifts.Scan(is,i,j) [L2,3,4,5] ... 573
20.3.8 PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i,j) [L2,3,4,5] ... 576
PF.Channlzn.* ... 578
20.3.9 PF.Channlzn.Descr.*m*(j) [L2,3,4,5] ... 578
20.3.10 PF.Channlzn.Disp.*m*(j) [L2,3,4,5] ... 580
20.3.11 PF.Channlzn.Hist.ChTrace.*m*(j,k) [L2,3,4,5] ... 582
20.3.12 PF.Channlzn.Hist.Comb.Code.*m*(j) [L2,3,4,5] ... 584
20.3.13 PF.Channlzn.Hist.Comb.Func.*m*(j) [L2,3,4,5] ... 587
20.3.14 PF.Channlzn.Hist.Comb.Alg.*m*(j) [L2,3,4,5] ... 589
20.3.15 PF.Channlzn.Hist.WF_Dat(j) [L2] ... 590
PF.CommonSigma.* ... 592
20.3.16 PF.CommonSigma.AggLevel [L2,3,4,5] ... 592
PF.Grid.* ... 593
20.3.17 PF.Grid.DelR [L2,3,4,5] ... 593
PF.Overlap.* ... 593
20.3.18 PF.Overlap.SubSwath.Dat [L2] ... 593
20.3.19 PF.Overlap.SubSwath.ID1(j) [L2] ... 595
20.3.20 PF.Overlap.SubSwath.ID2(j) [L2] ... 596
20.3.21 PF.Overlap.SubSwath.IDL1(j) [L2] ... 597
20.3.22 PF.Overlap.Subswath.IDL2(j) [L2] ... 598
PF.Point.* ... 599
20.3.23 PF.Point.IrayNum.*TileNum*(ipt, j) [L2] ... 599
20.3.24 PF.Point.SenFrmRefPulse(j_tile) [L2,3,4,5] ... 601
20.3.25 PF.Point.SynBspread.*TileNum*(ipt) [L2,3,4,5] ... 602
20.3.26 PF.Point.SynPspread.*TileNum* (ipt) [L2,3,4,5] ... 603
20.3.27 PF.Point.SynTasymtry.*TileNum* (ipt) [L2,3,4,5] ... 604
20.3.28 PF.Point.SynTspread.*TileNum* (ipt) [L2,3,4,5] ... 605
20.3.29 PF.Point.SynXspread.*TileNum* (ipt) [L2,3,4,5] ... 606
20.3.30 PF.Point.SynYspread.*TileNum* (ipt) [L2,3,4,5] ... 607
20.3.31 PF.Point.SynZspread.*TileNum* (ipt) [L2,3,4,5] ... 607
PF.Recon.* ... 608
20.3.32 PF.Recon.PrevProdNames(mm,k) [L2,3,4,5] ... 608
20.3.33 PF.Recon.App(j) [L1,2,3,4,5] ... 610
20.3.34 PF.Recon.AppDescr(j) [L1,2,3,4,5] ... 611
20.3.35 PF.Recon.CommandFiles(j) [L1,2,3,4,5] ... 612
20.3.36 PF.Recon.RootDS(j) [L2,3,4,5] ... 613
20.3.37 PF.Recon.Site(j) [L1,2,3,4,5] ... 616
PF.Registn.* ... 617

20.3.38 PF.Registn.PulseDat(j) [L2] ... 617
20.3.39 PF.Registn.ScanDat(j) [L2,3,4,5] ... 618
20.3.40 PF.Registn.SubswathDat(iswl,j) [L2,3,4,5] ... 620
PF.SAL.* ... 622
20.3.41 PF.SAL.Autofocus [L2,3,4,5] ... 622
20.3.42 PF.SAL.CollMode [L2,3,4,5] ... 622
20.3.43 PF.SAL.DOF [L2,3,4,5] ... 623
20.3.44 PF.SAL.ProcGrid.Axes.*ChNum.TileNum*(j) [L2,3,4,5] ... 623
20.3.45 PF.SAL.ProcGrid.Code [L2,3,4,5] ... 623
20.3.46 PF.SAL.ProcGrid.Normal.*ChNum.TileNum* (j) [L2,3,4,5] ... 623
20.3.47 PF.SAL.ProcGrid.GRP.*ChNum.TileNum*(j) [L2,3,4,5] ... 624
20.3.48 PF.SAL.ProcGrid.Type [L2,3,4,5] ... 624
PF.Sort.* ... 625
20.3.49 PF.Sort.PointOrder.*TileNum* [L2,3,4,5] ... 625
20.3.50 PF.Sort.PointPriority.*TileNum*(j) [L2,3,4,5] ... 625
20.3.51 PF.Sort.WFClpOrder.*TileNum* [L2] ... 626
20.3.52 PF.Sort.WFClpPriority.*TileNum*(j) [L2] ... 627
PF.Subswath.* ... 628
20.3.53 PF.Subswath.Nlayers[L2,3,4,5] ... 628
20.3.54 PF.Subswath.NumScans(iswl) [L2] ... 629
20.3.55 PF.Subswath.NumSubswaths(iswl) [L2] ... 630
20.3.56 PF.Subswath.ScansList.*iswl*(isw,it,j) [L2,3,4,5] ... 630
PF.Swath.* ... 633
20.3.57 PF.Swath.OffsetStatus [L2,3,4,5] ... 633
20.3.58 PF.Swath.SynBspread [L2,3,4,5] ... 634
20.3.59 PF.Swath.SynPspread [L2,3,4,5] ... 635
20.3.60 PF.Swath.SynTasymtry [L2,3,4,5] ... 635
20.3.61 PF.Swath.SynTspread [L2,3,4,5] ... 636
20.3.62 PF.Swath.SynXspread[L2,3,4,5] ... 636
20.3.63 PF.Swath.SynYspread[L2,3,4,5] ... 637
20.3.64 PF.Swath.SynZspread[L2,3,4,5] ... 637
PF.SynMethod.* ... 638
20.3.65 PF.SynMethod.Intensity [L2,3,4,5] ... 638
20.3.66 PF.SynMethod.Position [L2,3,4,5] ... 640
20.3.67 PF.SynMethod.Time[L2,3,4,5] ... 641
PF.TimeToRange.* ... 641
20.3.68 PF.TimeToRange.BendEffect [L2,3,4,5] ... 641
20.3.69 PF.TimeToRange.Model [L2,3,4,5] ... 642
20.3.70 PF.TimeToRange.Parms.Ht_Surface(j) [L1,2] ... 643
20.3.71 PF.TimeToRange.Parms.Hu_Surface(j) [L1,2] ... 643
20.3.72 PF.TimeToRange.Parms.Lat(j) [L1,2] ... 644
20.3.73 PF.TimeToRange.Parms.Nsurface(j) [L1,2] ... 644
20.3.74 PF.TimeToRange.Parms.P_Surface(j) [L1,2] ... 644
20.3.75 PF.TimeToRange.Parms.ScaleHeight(j) [L1,2] ... 645
20.3.76 PF.TimeToRange.Parms.T_Surface(j) [L1,2] ... 645
PF.UserDefined.* ... 646
20.3.77 PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) [L1,2,3,4,5] ... 646
20.4 PD.* PRODUCT DEFINITION ... 647
PD.DatDscrptors.* ... 647
20.4.1 PD.DatDscrptors.Uncerty(j) [L1,2,3,4,5] ... 647
PD.Hi/LoIndx.* ... 648
20.4.2 PD.HiIndx.AdjacentP(j) [L2,3,4,5] ... 648
20.4.3 PD.LoIndx.AdjacentP(j) [L2,3,4,5] ... 649
20.4.4 PD.HiIndx.AdjacentW(j) [L1,2,3,4,5] ... 649
20.4.5 PD.LoIndx.AdjacentW(j) [L1,2,3,4,5] ... 650
20.4.6 PD.HiIndx.Array.Elem(ia,j) [L1,2] ... 651
20.4.7 PD.LoIndx.Array.Elem(ia,j) [L1,2] ... 652
20.4.8 PD.HiIndx.Array(j) [L1,2] ... 653
20.4.9 PD.LoIndx.Array (j) [L1,2] ... 655
20.4.10 PD.HiIndx.Artifacts [L1,2,3,4,5] ... 656
20.4.11 PD.LoIndx.Artifacts [L1,2,3,4,5] ... 657

20.4.12 PD.HiIndx.Channlzn.ChRg(m) [L1,2,3,4,5] ... 657
20.4.13 PD.LoIndx.Channlzn.ChRg(m) [L1,2,3,4,5] ... 658
20.4.14 PD.HiIndx.Channlzn.Rg_m [L1,2,3,4,5] ... 658
20.4.15 PD.LoIndx.Channlzn.Rg_m [L1,2,3,4,5] ... 660
20.4.16 PD.HiIndx.DAL [L1,2,3,4,5] ... 661
20.4.17 PD.LoIndx.DAL [L1,2,3,4,5] ... 661
20.4.18 PD.HiIndx.Fellows[L1,2,3,4,5] ... 662
20.4.19 PD.LoIndx.Fellows[L1,2,3,4,5] ... 662
20.4.20 PD.HiIndx.IPP(j) [L1,2,3,4,5] ... 662
20.4.21 PD.LoIndx.IPP(j) [L1,2,3,4,5] ... 663
20.4.22 PD.HiIndx.Laser(j) [L1,2,3,4,5] ... 664
20.4.23 PD.LoIndx.Laser(j) [L1,2,3,4,5] ... 665
20.4.24 PD.HiIndx.MD_Dat [L1,2,3,4,5] ... 666
20.4.25 PD.LoIndx.MD_Dat [L1,2,3,4,5] ... 666
20.4.26 PD.HiIndx.MD_Strings [L1,2,3,4,5] ... 666
20.4.27 PD.LoIndx.MD_Strings [L1,2,3,4,5] ... 667
20.4.28 PD.HiIndx.Overlap(j) [L1,2] ... 668
20.4.29 PD.LoIndx.Overlap(j) [L1,2] ... 669
20.4.30 PD.HiIndx.Parents[L1,2,3,4,5] ... 669
20.4.31 PD.LoIndx.Parents [L1,2,3,4,5] ... 670
20.4.32 PD.HiIndx.Point.RangeListC.*OrdIndx* [L2,3,4,5] ... 671
20.4.33 PD.LoIndx.Point.RangeListC.*OrdIndx* [L2,3,4,5] ... 672
20.4.34 PD.HiIndx.Point.RangeListX.*OrdIndx* [L2,3,4,5] ... 672
20.4.35 PD.LoIndx.Point.RangeListX.*OrdIndx*[L2,3,4,5] ... 673
20.4.36 PD.HiIndx.Point.SigmaDat [L2,3,4,5] ... 674
20.4.37 PD.LoIndx.Point.SigmaDat [L2,3,4,5] ... 675
20.4.38 PD.HiIndx.Point.Tile [L2,3,4,5] ... 675
20.4.39 PD.LoIndx.Point.Tile [L2,3,4,5] ... 676
20.4.40 PD.HiIndx.PrevStages(k) [L2,3,4,5] ... 676
20.4.41 PD.LoIndx.PrevStages(k) [L2,3,4,5] ... 677
20.4.42 PD.HiIndx.Pulse [L1,2,3,4,5] ... 677
20.4.43 PD.LoIndx.Pulse [L1,2,3,4,5] ... 678
20.4.44 PD.HiIndx.Quench[L1,2] ... 678
20.4.45 PD.LoIndx.Quench[L1,2] ... 679
20.4.46 PD.HiIndx.Ray.*TileNum* [L1,2] ... 679
20.4.47 PD.LoIndx.Ray.*TileNum*[L1,2] ... 680
20.4.48 PD.HiIndx.Recon(j) [L1,2,3,4,5] ... 680
20.4.49 PD.LoIndx.Recon(j) [L1,2,3,4,5] ... 681
20.4.50 PD.HiIndx.Remap [L2,3,4,5] ... 682
20.4.51 PD.LoIndx.Remap [L2,3,4,5] ... 683
20.4.52 PD.HiIndx.RgAmb(j) [L1,2,3,4,5] ... 683
20.4.53 PD.LoIndx.RgAmb(j) [L1,2,3,4,5] ... 684
20.4.54 PD.HiIndx.Scan (j) [L1,2,3,4,5] ... 684
20.4.55 PD.LoIndx.Scan(j) [L1,2,3,4,5] ... 685
20.4.56 PD.HiIndx.SubSwAdj(iswl) [L2,3,4,5] ... 686
20.4.57 PD.LoIndx.SubSwAdj(iswl) [L2,3,4,5] ... 687
20.4.58 PD.HiIndx.SubSwScGrpClasses(iswl) [L2,3,4,5] ... 687
20.4.59 PD.LoIndx.SubSwScGrpClasses(iswl) [L2,3,4,5] ... 688
20.4.60 PD.HiIndx.Swath(j) [L1,2,3,4,5] ... 689
20.4.61 PD.LoIndx.Swath(j) [L1,2,3,4,5] ... 690
20.4.62 PD.HiIndx.TileMap.DedictdTile [L1,2,3,4,5] ... 691
20.4.63 PD.LoIndx.TileMap.DedictdTile [L1,2,3,4,5] ... 692
20.4.64 PD.HiIndx.Tsegts.*TileNum* [L2,3,4,5] ... 693
20.4.65 PD.LoIndx.Tsegts.*TileNum* [L2,3,4,5] ... 693
20.4.66 PD.HiIndx.Truth(j) [L1,2,3,4,5] ... 694
20.4.67 PD.LoIndx.Truth(j) [L1,2,3,4,5] ... 694
20.4.68 PD.HiIndx.UDNB*n* [L1,2,3,4,5] ... 695
20.4.69 PD.LoIndx.UDNB*n* [L1,2,3,4,5] ... 696
20.4.70 PD.HiIndx.Voids(j) [L1,2,3] ... 696
20.4.71 PD.LoIndx.Voids(j) [L1,2,3] ... 697
20.4.72 PD.HiIndx.WF(j) [L1,2] ... 698
20.4.73 PD.LoIndx.WF(j) [L1,2] ... 698

20.4.74 PD.HiIndx.WFClpPulseNum.*TileNum* [L2] 699
20.4.75 PD.LoIndx.WFClpPulseNum.*TileNum*[L2] 700
20.4.76 PD.HiIndx.Zones [L1,2,3,4,5] 700
20.4.77 PD.LoIndx.Zones [L1,2,3,4,5] 701
PD.Point.* 701
20.4.78 PD.Point.RangeListC.*OrdIndx*(j,k) [L2,3,4,5] 702
20.4.79 PD.Point.RangeListX.*OrdIndx*(j,k) [L2,3,4,5] 704
PD.Sort.* 706
20.4.80 PD.Sort.NativeOrder [L1,2,3,4,5] 706
20.4.81 PD.Sort.NativePri(j) [L1,2,3,4,5] 707
20.4.82 PD.Sort.OrderFlag(j_tile) [L2,3,4,5] 709
20.4.83 PD.Sort.OrderVec.*TileNum*(j) [L2,3,4,5] 709
PD.TileMap.* 710
20.4.84 PD.TileMap.AdjacencyPts(n_tile) [L2,3,4,5] 710
20.4.85 PD.TileMap.AdjacencyWFs(n_tile) [L1,2,5] 713
20.4.86 PD.TileMap.ChanList(j_tile,k) [L1,2,3,4,5] 716
20.4.87 PD.TileMap.DedictdTile.Cnstrnt1.Name [L1,2,3,4,5] 717
20.4.88 PD.TileMap.DedictdTile.Cnstrnt1.Val(j) [L1,2,3,4,5] 718
20.4.89 PD.TileMap.DedictdTile.Cnstrnt2.Name [L1,2,3,4,5] 720
20.4.90 PD.TileMap.DedictdTile.Cnstrnt2.Val(j) [L1,2,3,4,5] 720
20.4.91 PD.TileMap.DedictdTile.Cnstrnt3.Name [L1,2,3,4,5] 722
20.4.92 PD.TileMap.DedictdTile.Cnstrnt3.Val(j) [L1,2,3,4,5] 723
20.4.93 PD.TileMap.DedictdTile.TileNumValHi(j) [L1,2,3,4,5] 724
20.4.94 PD.TileMap.DedictdTile.TileNumValLow(j) [L1,2,3,4,5] 725
20.4.95 PD.TileMap.GlblPtTyping [L2,3,4,5] 726
20.4.96 PD.TileMap.Lat(j_tile,k) [L1, 2,3,4,5] 726
20.4.97 PD.TileMap.Lon(j_tile,k) [L1,2,3,4,5] 728
20.4.98 PD.TileMap.MaxNsamps.*TileNum* [L1,2] 729
20.4.99 PD.TileMap.N_Rows(j_tile) [L2,3,4,5] 730
20.4.100 PD.TileMap.N_Tiles [L1,2,3,4,5] 731
20.4.101 PD.TileMap.PtType(j_tile) [L2,3,4,5] 732
20.4.102 PD.TileMap.PulseRange(j_tile,k) [L1,2,3,4,5] 732
20.4.103 PD.TileMap.RayRange(j_tile,k) [L1,2] 734
20.4.104 PD.TileMap.Return_Range(j_tile,k) [L2,3,4,5] 735
20.4.105 PD.TileMap.StartPulseNum.*TileNum*(j_parent, j,k) [L2,3,4,5] 735
20.4.106 PD.TileMap.TileDelT [L1,2,3,4,5] 740
20.4.107 PD.TileMap.TileNum(j_tile) [L1,2,3,4,5] 741
20.4.108 PD.TileMap.MGRS_Zone(j_tile) [L1, 2,3,4,5] 742
20.4.109 PD.TileMap.X_Range(j_tile,k) [L2,3,4,5] 744
20.4.110 PD.TileMap.Y_Range(j_tile,k) [L2,3,4,5] 745
20.4.111 PD.TileMap.Z_Range(j_tile,k) [L2,3,4,5] 746
20.4.112 PD.TileMap.T_Range(j_tile,k) [L1,2,3,4,5] 746
20.4.113 PD.TileMap.WF_Returns(ich,j) [L2] 747
PD.IntTileMap.* 748
20.4.114 PD.IntTileMap.RtnsCode.*TileNum*.Flg [L2,3,4,5] 748
20.4.115 PD.IntTileMap.PulseNum.*TileNum*.Flg [L2,3,4,5] 749
20.4.116 PD.IntTileMap.ArrayNum.*TileNum*.Flg [L2,3,4,5] 749
20.4.117 PD.IntTileMap.ChNum.*TileNum*.Flg [L2,3,4,5] 750
20.4.118 PD.IntTileMap.DirCase.*TileNum*.Flg [L1,2] 750
20.4.119 PD.IntTileMap.PolCode.*TileNum*.Flg [L1,2,3,4,5] 751
20.4.120 PD.IntTileMap.FilterCode.*TileNum*.Flg [L1,2,3,4,5] 752
20.4.121 PD.IntTileMap.RecordStrt.*TileNum*.Flg [L1,2] 752
20.4.122 PD.IntTileMap.Scan.Num.*TileNum*.Flg [L1,2] 753
20.4.123 PD.IntTileMap.Scan.Phase.*TileNum*.Flg [L1,2] 753
20.4.124 PD.IntTileMap.NadirClass.*ChNum*.Flg [L1,2,3,4,5] 754
20.4.125 PD.IntTileMap.NadirClass.*ChNum*.Reps(j) [L1,2,3,4,5] 755
20.4.126 PD.IntTileMap.Origin.E_X.Flg [L2,3,4,5] 756
20.4.127 PD.IntTileMap.Origin.E_Y.Flg [L2,3,4,5] 756
20.4.128 PD.IntTileMap.Origin.E_Z.Flg [L2,3,4,5] 757
20.4.129 PD.IntTileMap.Origin.E_X.Reps(j) [L2,3,4,5] 757
20.4.130 PD.IntTileMap.Origin.E_Y.Reps(j) [L2,3,4,5] 758
20.4.131 PD.IntTileMap.Origin.E_Z.Reps(j) [L2,3,4,5] 759

20.4.132 PD.IntTileMap.Origin.Z_X.Flg [L2,3,4,5] 759
20.4.133 PD.IntTileMap.Origin.Z_Y.Flg [L2,3,4,5] 760
20.4.134 PD.IntTileMap.Origin.Z_Z.Flg [L2,3,4,5] 761
20.4.135 PD.IntTileMap.Origin.Z_X.Reps(j) [L2,3,4,5] 761
20.4.136 PD.IntTileMap.Origin.Z_Y.Reps(j) [L2,3,4,5] 762
20.4.137 PD.IntTileMap.Origin.Z_Z.Reps(j) [L2,3,4,5] 763
20.4.138 PD.IntTileMap.Pos.*TileNum*.Flg [L2,3,4,5] 763
20.4.139 PD.IntTileMap.Intensity.*TileNum*.Flg [L2,3,4,5] 764
20.4.140 PD.IntTileMap.Nsamps.*TileNum*.Flg [L1,2] 764
20.4.141 PD.IntTileMap.Snstvty.*ChNum*.*TileNum*.Flg [L1,2] 765
20.4.142 PD.IntTileMap.Snstvty.Reps.*ChNum*.*TileNum*(j) [L1,2] 766
20.4.143 PD.IntTileMap.TruthTags.*TileNum*.Flg [L2,3,4,5] 768
20.4.144 PD.IntTileMap.Tstart.*TileNum*.Flg [L1,2] 769
20.4.145 PD.IntTileMap.TxCount.*TileNum*.Flg [L1,2,3,4,5] 769
PD.UserDefined.* 770
20.4.146 PD.UserDefined.NB_Def.P*n* [L1,2,3,4,5] 770
20.4.147 PD.UserDefined.NB_Type.P*n*[L1,2,3,4,5] 770
20.4.148 PD.UserDefined.PtDef.P*n*[L2,3,4,5] 771
20.4.149 PD.UserDefined.PtType.P*n*[L2,3,4,5] 772
20.4.150 PD.UserDefined.PulseDef.P*n*[L1,2,3,4,5] 772
20.4.151 PD.UserDefined.PulseType.P*n*[L1,2,3,4,5] 773
PD.WFClp.* 773
20.4.152 PD.WFClp.Nsamps.*TileNum*(iray/irays, ipi/j) [L2] 774
20.4.153 PD.WFClp.PulseNum.*TileNum*(ipi) [L2] 778
PD.Wordsize.* 780
20.4.154 PD.Wordsize.ChCombFunc [L2,L3] 780
20.4.155 PD.Wordsize.InfAssur [L1,2,3,4,5] 781
20.4.156 PD.Wordsize.Iray [L1,2] 781
20.4.157 PD.Wordsize.Intnsy [L2,3,4,5] 782
20.4.158 PD.Wordsize.Meta(j) [L1,2,3,4,5] 782
20.4.159 PD.Wordsize.Pos [L2,3,4,5] 783
20.4.160 PD.Wordsize.SensorPos [L1,2,3,4,5] 783
20.4.161 PD.Wordsize.Pulse [L1,2,3,4,5] 784
20.4.162 PD.Wordsize.Recon [L1,2,3,4,5] 785
20.4.163 PD.Wordsize.RxDatVec [L1,2] 785
20.4.164 PD.Wordsize.TileNum [L1,2,3,4,5] 786
20.4.165 PD.Wordsize.NB_Def.P*n*[L1,2,3,4,5] 786
20.4.166 PD.Wordsize.PtDef.P*n* [L2,3,4,5] 787
20.4.167 PD.Wordsize.PulseDef.P*n* [L1,2,3,4,5] 787
20.4.168 PD.Wordsize.Truth(j) [L1,2,3,4,5] 788
20.4.169 PD.Wordsize.TxDat [L1,2,3] 788
20.4.170 PD.Wordsize.Zones [L1,2,3,4,5] 789
20.5 SC.* SENSOR CONFIGURATION 791
SC.Array.* 791
20.5.1 SC.Array.BadElmnts.*Ia*(j) [L1,2] 791
20.5.2 SC.Array.CrossTalkQ1.*Ia*(j,k,n) [L1,2] 792
20.5.3 SC.Array.CrossTalkQ2.*Ia*(j,k,n) [L1,2] 794
20.5.4 SC.Array.CrossTalkQ3.*Ia*(j,k,n) [L1,2] 796
20.5.5 SC.Array.CrossTalkQ4.*Ia*(j,k,n) [L1,2] 798
20.5.6 SC.Array.FilterDat(ifc,j) [L1,2,3,4,5] 799
20.5.7 SC.Array.OptclCent(ia,j) [L1,2] 800
20.5.8 SC.Array.PolDat(ipc,j) [L1,2,3,4,5] 802
20.5.9 SC.Array.RayDirs.*Ia*(icase,ie,j) [L1,2] 806
20.5.10 SC.Array.RayDirsSigmas(ia,icase,j) [L1,2] 808
20.5.11 SC.Array.LatticeDesc(ia,j) [L1,2] 809
20.5.12 SC.Array.Chunks(ia,ichk,j) [L1,2] 810
20.5.13 SC.Array.CurrentResp.Dat(ia,j) [L1,2] 811
20.5.14 SC.Array.CurrentResp.Flg(ia) [L1,2] 813
20.5.15 SC.Array.DirCaseNum.*TileNum*(ip,j) [L1,2] 814
SC.Artifacts.* 815
20.5.16 SC.Artifacts.List(j) [L1,2,3,4,5] 816

20.5.17 SC.Artifacts.Descr(j) [L1,2,3,4,5] ... 816
SC.Channlzn.* ... 817
20.5.18 SC.Channlzn.Descr(j,k) [L1,2,3,4,5] ... 817
20.5.19 SC.Channlzn.Disp(j) [L1,2,3,4,5] ... 818
SC.Laser.* ... 819
20.5.20 SC.Laser.Dat(il,j) [L1,2,3,4,5] ... 819
20.5.21 SC.Laser.IDat(il,j) [L1,2,3,4,5] ... 820
20.5.22 SC.Laser.Beam(il, i, j) [L1,2] ... 823
20.5.23 SC.Laser.DelCosAlpha [L1,2] ... 826
20.5.24 SC.Laser.DelCosBeta[L1,2] ... 826
SC.MultiDwell.* ... 827
20.5.25 SC.MultiDwell.Dat(j) [L1,2,3,4,5] ... 827
20.5.26 SC.MultiDwell.Flg[L1,2,3,4,5] ... 827
SC.Overlap.* ... 828
20.5.27 SC.Overlap.PulsePrcnt[L1,2] ... 828
20.5.28 SC.Overlap.Pulse.X_Hi(j,k) [L1,2] ... 829
20.5.29 SC.Overlap.Pulse.X_Lo(j,k) [L1,2] ... 831
20.5.30 SC.Overlap.Pulse.Y_Hi(j,k) [L1,2] ... 832
20.5.31 SC.Overlap.Pulse.Y_Lo(j,k) [L1,2] ... 833
20.5.32 SC.Overlap.ScanPrcnt [L1,2] ... 834
20.5.33 SC.Overlap.Scan.ID1(j) [L1,2] ... 834
20.5.34 SC.Overlap.Scan.ID2(j) [L1,2] ... 835
20.5.35 SC.Overlap.Scan.X_Hi(j) [L1,2] ... 836
20.5.36 SC.Overlap.Scan.X_Lo(j) [L1,2] ... 837
20.5.37 SC.Overlap.Scan.Y_Hi(j) [L1,2] ... 838
20.5.38 SC.Overlap.Scan.Y_Lo(j) [L1,2] ... 839
20.5.39 SC.Overlap.SubswathPrcnt [L1,2] ... 840
SC.Pulse.* ... 841
20.5.40 SC.Pulse.ArrayMajorAxis.*TileNum*(ia, ip, j) [L1,2] ... 841
20.5.41 SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) [L1,2] ... 843
20.5.42 SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j) [L1,2] ... 844
20.5.43 SC.Pulse.BmMinorAxis.*TileNum* (il,ip,j) [L1,2] ... 846
20.5.44 SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip, ie, k) [L1,2] ... 847
20.5.45 SC.Pulse.FilterCode.*TileNum*(ip, j) [L1,2,3,4,5] ... 851
20.5.46 SC.Pulse.IllumntrPos.*ChNum.TileNum*(ip,j) [L1,2,3,4,5] ... 852
20.5.47 SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) [L1,2,3,4,5] ... 854
20.5.48 SC.Pulse.PolCode.*TileNum*(ip, j) [L1,2,3,4,5] ... 855
20.5.49 SC.Pulse.Pos.X_Offset(j_tile) [L1,2,3,4,5] ... 856
20.5.50 SC.Pulse.Pos.Y_Offset(j_tile) [L1,2,3,4,5] ... 857
20.5.51 SC.Pulse.Pos.Z_Offset(j_tile) [L1,2,3,4,5] ... 858
20.5.52 SC.Pulse.RecordStrt.*TileNum*(ip, j) [L1,2] ... 859
20.5.53 SC.Pulse.Scan.Num.*TileNum*(ip, j) [L1,2] ... 860
20.5.54 SC.Pulse.Scan.Phase.*TileNum*(ip, j) [L1,2] ... 861
20.5.55 SC.Pulse.SensorAxisX.*TileNum*(ip,j) [L1,2] ... 863
20.5.56 SC.Pulse.SensorAxisY.*TileNum*(ip,j) [L1,2] ... 864
20.5.57 SC.Pulse.SensorPos.*ChNum.TileNum*(ip,j) [L1,2,3,4,5] ... 864
20.5.58 SC.Pulse.TxCount.*Il.TileNum*(ip,j) [L1,2,3,4,5] ... 867
20.5.59 SC.Pulse.TxDelay.*TileNum*(ip,il) [L1,2] ... 870
20.5.60 SC.Pulse.TxPos.X_Offset(j_tile) [L1,2,3,4,5] ... 872
20.5.61 SC.Pulse.TxPos.Y_Offset(j_tile) [L1,2,3,4,5] ... 873
20.5.62 SC.Pulse.TxPos.Z_Offset(j_tile) [L1,2,3,4,5] ... 873
SC.Quench.* ... 874
20.5.63 SC.Quench.Times(j) [L1,2,3,4,5] ... 874
20.5.64 SC.Quench.Outage [L1,2,3,4,5] ... 875
SC.RgAmb.* ... 876
20.5.65 SC.RgAmb.Num.*Il*(isnc,j) [L1,2,3,4,5] ... 876
20.5.66 SC.RgAmb.StratFlg(j) [L1,2,3,4,5] ... 878
SC.Scan.* ... 885
20.5.67 SC.Scan.Axes.ThetaMaj(iscp, ich) [L1,2] ... 885
20.5.68 SC.Scan.Axes.ThetaMin(iscp,ich) [L1,2] ... 886
20.5.69 SC.Scan.ConePulseTable(isci) [L1,2] ... 887

20.5.70 SC.Scan.N_Major(iscp,ich,k) [L1,2] 888
20.5.71 SC.Scan.NomGroundLen(iscp) [L1,2,3,4,5] 889
20.5.72 SC.Scan.NadirClass.*ChNum*(is) [L1,2] 890
20.5.73 SC.Scan.NomNadir.Max(isnc) [L1,2] 892
20.5.74 SC.Scan.NomNadir.Mean(isnc) [L1,2,3,4,5] 893
20.5.75 SC.Scan.NomNadir.Min(isnc) [L1,2,3,4,5] 894
20.5.76 SC.Scan.PhaseBins(isnc,j) [L1,2,3] 894
20.5.77 SC.Scan.Type[L1,2,3,4,5] 896
20.5.78 SC.Scan.U_Cone(iscp, ich, k) [L1,2] 897
SC.Subswath.* 898
20.5.79 SC.Subswath.TotNum [L1,2] 898
20.5.80 SC.Subswath.NumScansPer [L1,2] 899
SC.Swath.* 900
20.5.81 SC.Swath.FineTimeCal.*Ia*(ie,j) [L1,2] 900
20.5.82 SC.Swath.MeanPhtnFlx.Dat(j) [L1,2] 901
20.5.83 SC.Swath.MeanPhtnFlx.Idat(j) [L1,2] 902
20.5.84 SC.Swath.Pos.X_Fac [L1,2,3,4,5] 903
20.5.85 SC.Swath.Pos.Y_Fac [L1,2,3,4,5] 903
20.5.86 SC.Swath.Pos.Z_Fac [L1,2,3,4,5] 904
20.5.87 SC.Swath.RateFactor(ia) [L1,2] 904
20.5.88 SC.Swath.RecordDelay[L1,2] 905
20.5.89 SC.Swath.SigmaClkStep [L1,2] 906
20.5.90 SC.Swath.TimeStep [L1,2] 907
20.5.91 SC.Swath.TimeSync[L1,2] 908
SC.WF.* 908
20.5.92 SC.WF.RayDesc.ArryGlbl(iray) [L1,2] 908
20.5.93 SC.WF.RayDesc.ElmtGlbl(iray) [L1,2] 909
20.5.94 SC.WF.RayDesc.Arry.*TileNum*(iray) [L1,2] 911
20.5.95 SC.WF.RayDesc.Elmt.*TileNum*(iray) [L1,2] 912
20.5.96 SC.WF.Rx.Nsamps(ich) [L1,2] 914
20.5.97 SC.WF.Rx.IntnsyUncty(ich) [L1,2,3,4,5] 914
20.5.98 SC.WF.TxProfile.FM_Slope(il) [L1,2] 915
20.5.99 SC.WF.TxProfile.FM_Start(il) [L1,2] 916
20.5.100 SC.WF.TxProfile.FM_PhaseStart(il) [L1,2] 917
20.5.101 SC.WF.TxProfile.Dat(il,j) [L1,2] 917
20.5.102 SC.WF.TxProfile.Flags(j) [L1,2] 919
20.5.103 SC.WF.TxProfile.Nsamps(il) [L1,2] 920
20.5.104 SC.WF.TxProfile.Peak(il) [L1,2] 920
20.5.105 SC.WF.VQ_Decode(ich, icode) [L1,2] 920
20.6 LD.* LIDAR DATA 921
LD.Point.* 921
20.6.1 LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) [L2] 922
20.6.2 LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) [L2] 925
20.6.3 LD.SAL.DataGrid.*ChNum*.*TileNum*(i,j,k) [L2] 927
LD.WFClp.* 929
20.6.4 LD.WFClp.RxDatVec.*TileNum*(ipi,iray, it) [L1,2] 929
20.6.5 LD.WFClp.Tstart.*TileNum*(iray/irays, ipi/j) [L1,2] 933
20.7 UD.* USER DEFINED WIDE BAND DATA 937
20.7.1 UD.PtDat.P*n*.*TileNum*(ipt) [L2,3,4,5] 937
20.7.2 UD.PulseDat.P*n*.*TileNum*(ipi) [L1,2,3,4,5] 938
20.7.3 *Eventually* UD.WaveDat.Pn.TileNum(ip,iray) [L1,2] 939

ANNEX A , ACRONYMS 940

ANNEX B , SAMPLE PROFILE 942

Figures

FIGURE 2- 1: CMMD, DAL AND CMMD COMPLIANT, DAL MANAGED DATA SETS....6

FIGURE 7- 1: --CONCEPTUAL MODEL STRUCTURE DIAGRAM....26

FIGURE 8- 1: THE LIDAR CONCEPTUAL MODEL IS A COMPOSITE OF A LIDAR PHYSICAL MODEL AND A DATA SET DOCUMENTATION MODEL....29

FIGURE 8.1- 1 LOOK DIRECTION ASSOCIATED WITH ARRAY AND ELEMENT CHOICE....30

FIGURE 8.2- 1: OVERVIEW OF THE LIDAR CONCEPTUAL MODEL DATA DOCUMENTATION STRUCTURE....32

FIGURE 9.1- 1: MD -TRUE METADATA – SECOND LEVEL (CLASS) STRUCTURE....34
FIGURE 9.1- 2: MD.COLLECT METADATA ELEMENTS....34
FIGURE 9.1- 3: MD.SENSOR METADATA ELEMENTS....34
FIGURE 9.1- 4: MD.PLATFORM METADATA ELEMENTS....35
FIGURE 9.1- 5: MD.RANGE METADATA ELEMENTS....35
FIGURE 9.1- 6: MD.DYNAMICS METADATA ELEMENTS....35
FIGURE 9.1- 7: MD.NOMGEO METADATA ELEMENTS....35
FIGURE 9.1- 8: MD.COVERAGE METADATA ELEMENTS....35
FIGURE 9.1- 9: MD.AGGREGATE METADATA ELEMENTS....36
FIGURE 9.1- 10: OBJECTIVE ELEMENTS FOR METADATA....36
FIGURE 9.1- 11: QUAL ELEMENTS FOR METADATA....36
FIGURE 9.1- 12: CANDIDATE INFORMATION ASSURANCE ELEMENTS FOR METADATA....36
FIGURE 9.1- 13: PROCHIST ELEMENTS FOR METADATA....37
FIGURE 9.1- 14: LEGALCONST ELEMENTS FOR METADATA....37
FIGURE 9.1- 15: META ELEMENTS FOR METADATA....37
FIGURE 9.1- 16: FILE ELEMENTS FOR METADATA....37
FIGURE 9.1- 17: TRUTH ELEMENTS FOR METADATA....38
FIGURE 9.1- 18: USEHIST ELEMENTS FOR METADATA....38

FIGURE 9.2- 1 ES – EXPLOITATION SUPPORT SECOND LEVEL DATA STRUCTURE....45
FIGURE 9.2- 2: ALIGN ELEMENTS FOR EXPLOITATION SUPPORT DATA....45
FIGURE 9.2- 3: COLLCONDN ELEMENTS FOR EXPLOITATION SUPPORT DATA....46
FIGURE 9.2- 4: PDOP ELEMENTS FOR EXPLOITATION SUPPORT DATA....46
FIGURE 9.2- 5: SCALE ELEMENTS FOR EXPLOITATION SUPPORT DATA....46
FIGURE 9.2- 6: OFFSET ELEMENTS FOR EXPLOITATION SUPPORT DATA....46
FIGURE 9.2- 7: ORIGIN ELEMENTS FOR EXPLOITATION SUPPORT DATA....46
FIGURE 9.2- 8: POINT ELEMENTS FOR EXPLOITATION SUPPORT DATA....46
FIGURE 9.2- 9: SCAN ELEMENTS FOR EXPLOITATION SUPPORT DATA....47
FIGURE 9.2- 10: SWATH ELEMENTS FOR EXPLOITATION SUPPORT DATA....47
FIGURE 9.2- 11: TRUTH ELEMENT FOR EXPLOITATION SUPPORT DATA....47
FIGURE 9.2- 12: WAVEFORM ELEMENT FOR EXPLOITATION SUPPORT DATA....47

FIGURE 9.3- 1: PF PRODUCT FORMATION SECOND LEVEL DATA STRUCTURE....64
FIGURE 9.3- 2: ADJUSTMENTS ELEMENTS FOR PRODUCT FORMATION....64
FIGURE 9.3- 3: RECONSTRUCTION ELEMENTS FOR PRODUCT FORMATION....64
FIGURE 9.3- 4: SUBSWATH ELEMENTS FOR PRODUCT FORMATION....65
FIGURE 9.3- 5: SYNMETHOD ELEMENTS FOR PRODUCT FORMATION....65
FIGURE 9.3- 6: USER DEFINED ELEMENTS FOR PRODUCT FORMATION....65
FIGURE 9.3- 7: POINT ELEMENTS FOR PRODUCT FORMATION....65
FIGURE 9.3- 8: SORT ELEMENTS FOR PRODUCT FORMATION....65
FIGURE 9.3- 9: SWATH ELEMENTS FOR PRODUCT FORMATION....65
FIGURE 9.3- 10: TIME TO RANGE ELEMENTS FOR PRODUCT FORMATION....66
FIGURE 9.3- 11: COMMON SIGMA ELEMENTS FOR PRODUCT FORMATION....66

FIGURE 9.3- 12: OVERLAP ELEMENTS FOR PRODUCT FORMATION....66
FIGURE 9.3- 13: CHANNELIZATION ON ELEMENTS FOR 67
FIGURE 9.3- 14: REGISTRATION ELEMENTS FOR PRODUCT FORMATION 67
FIGURE 9.3- 15: GRID ELEMENT FOR PRODUCT FORMATION 67

FIGURE 9.4- 1: PRODUCT DEFINITION SECOND LEVEL DATA 74
FIGURE 9.4- 2: LOINDX AND HIINDX ELEMENTS FOR PRODUCT DEFINITION 75
FIGURE 9.4- 3: SORT ELEMENTS FOR PRODUCT DEFINITION....75
FIGURE 9.4- 4: PULSE ELEMENTS FOR PRODUCT DEFINITION....75
FIGURE 9.4- 5: TILEMAP ELEMENTS FOR PRODUCT DEFINITION....76
FIGURE 9.4- 6: INTERNAL TILEMAP ELEMENTS FOR PRODUCT DEFINITION....76
FIGURE 9.4- 7: POINT ELEMENTS FOR PRODUCT DEFINITION 77
FIGURE 9.4- 8: USER DEFINED ELEMENTS FOR PRODUCT DEFINITION....77
FIGURE 9.4- 9: WORD SIZE ELEMENTS FOR PRODUCT DEFINITION 77
FIGURE 9.4- 10: WFCLP ELEMENTS FOR PRODUCT DEFINITION....77
FIGURE 9.4- 11: DATDSCRPTORS ELEMENTS FOR PRODUCT DEFINITION 78

FIGURE 9.5- 1: SC.* SENSOR CONFIGURATION SECOND LEVEL DATA STRUCTURE 90
FIGURE 9.5- 2: ARRAY ELEMENTS FOR SENSOR CONFIGURATION 91
FIGURE 9.5- 3: SCAN ELEMENTS FOR SENSOR CONFIGURATION....91
FIGURE 9.5- 4: RGAMB ELEMENTS FOR SENSOR CONFIGURATION....91
FIGURE 9.5- 5: SUBSWATH ELEMENTS FOR SENSOR CONFIGURATION 91
FIGURE 9.5- 6: SWATH ELEMENTS FOR SENSOR CONFIGURATION 91
FIGURE 9.5- 7: OVERLAP ELEMENTS FOR SENSOR CONFIGURATION....91
FIGURE 9.5- 8: PULSE ELEMENTS FOR SENSOR CONFIGURATION 92
FIGURE 9.5- 9: ARTIFACT ELEMENTS FOR SENSOR CONFIGURATION....92
FIGURE 9.5- 10: QUENCH ELEMENTS FOR SENSOR CONFIGURATION....92
FIGURE 9.5- 11: CHANNELZN ELEMENTS FOR SENSOR CONFIGURATION....92
FIGURE 9.5- 12: MULTIDWELL ELEMENTS FOR SENSOR CONFIGURATION....92
FIGURE 9.5- 13: WF ELEMENTS FOR SENSOR CONFIGURATION 93
FIGURE 9.5- 14: LASER ELEMENTS FOR SENSOR CONFIGURATION....93

FIGURE 9.6- 1: LD.* LIDAR DATA SECOND LEVEL DATA STRUCTURE 102
FIGURE 9.6- 2: POINT ELEMENTS FOR LIDAR DATA 102
FIGURE 9.6- 3: WFCLP ELEMENTS FOR LIDAR DATA....102

FIGURE 9.7- 1: UD.* USER DEFINED SECOND LEVEL DATA STRUCTURE 103
FIGURE 9.7- 2: UD.* USER DEFINED POINT ORIENTED DATA STRUCTURE....104
FIGURE 9.7- 3: UD.* USER DEFINED PULSE ORIENTED DATA ELELMENTS....104
FIGURE 9.7- 4: UD.* USER DEFINED WAVE ORIENTED DATA ELEMENTS 104

FIGURE 10.1.1- 1 SLOW TIME INTERPULSE PERIOD SEQUENCE ILLUSTRATION 110
FIGURE 10.1.1- 2 FAST TIME INTERPULSE PERIOD STRUCTURE ILLUSTRATION 111
FIGURE 10.1.1- 3 DISCRETE TIME INTERVAL CONVENTION FOR DETERMINING THE TIME STEP THAT IS THE END OF AN INTERVAL GIVEN ITS START AND ITS DURATION. 112

FIGURE 10.1.2- 1 EFFECTIVE POSITION AND ORIENTATION OF ARRAY AND BEAM PLANES 122
FIGURE 10.1.2- 2 DIRECTION COSINES RELATIVE TO BEAM AND ARRAY PLANES 122
FIGURE 10.1.2- 3 SCAN CONE AXIS DESCRIBED BY DIRECTION COSINES. 126
FIGURE 10.1.2- 4 SCANNING ON AN ELLIPTICAL CONE PARAMETERIZED BY SCAN PHASE....127

FIGURE 10.1.4- 1 HIERARCHICAL RELATIONSHIP BETWEEN PULSES, SCANS, SUBSWATHS AND THE PRODUCT LEVEL SWATH FOR LIDAR DATA SETS 137
FIGURE 10.1.4- 2 PCD VISUALIZATION 140

FIGURE 10.1.4- 3 EXAMPLE CROSS-TRACK SCANS IN SUBSWATHS. A) SENSOR FRAME SERPENTINE, B) SENSOR FRAME CARRIAGE RETURN (HALF CYCLE), C) EARTH FRAME SERPENTINE, D) EARTH FRAME CARRIAGE RETURN 146
FIGURE 10.1.4- 4 EXAMPLE ALONG-TRACK SCANS IN COLLECTION SUBSWATHS. A) SENSOR FRAME CARRIAGE RETURN, B) SENSOR FRAME SERPENTINE, C) EARTH FRAME CARRIAGE RETURN, D) EARTH FRAME SERPENTINE 146
FIGURE 10.1.4- 5 SCAN OVERLAP LEVELS AND POSSIBLE PURPOSES 149
FIGURE 10.1.4- 6 TRACE OF ADJUSTMENTS DERIVED FROM REGISTRATIONS USED TO COMPOSE SCANS, SUBSWATHS AND THE FINAL SWATH. 152

FIGURE 10.1.5- 1 ERROR MODEL OUTPUTS FOR SINGLE AND MULTIPLE POINT ERROR PROPAGATION 155
FIGURE 10.1.5- 2 A 7 X 7 SENSOR ERROR COVARIANCE MATRIX IS THE BASIS FOR RIGOROUS LIDAR ERROR PROPAGATION. 155

FIGURE 10.2.1- 1 TILING'S MAIN FEATURES DESCRIBED AT TOP LEVEL 169
FIGURE 10.2.1- 2 TWO RETURN ASSIGNMENTS FOR A SINGLE PCD TRACE 172
FIGURE 10.2.1- 3 TILING OVERVIEW, POINT DATA ILLUSTRATION 173
FIGURE 10.2.1- 4 "ROW" MODEL OF POINT, PULSE AND ARRAY [DETERMINED BY THE CHANNEL DESCRIPTION] ASSOCIATED DATA GROUPS 176
FIGURE 10.2.1- 5 "ROW" MODEL OF WAVEFORM, PULSE AND ARRAY ASSOCIATED DATA GROUPS 177
FIGURE 10.2.1- 6 OVERLAP IS NEEDED TO AVOID INTRODUCING VOIDS WHEN TILING TO LOCAL GEODETIC GEOMETRY 179
FIGURE 10.2.1- 7 ILLUSTRATION OF DOCUMENTATION OF DEDICATED TILE GROUPS. EACH ROW OF THE TABLE DESCRIBES A TILE GROUP. EACH COLUMN IS PROVIDED BY A PD.TILEMAP ITEM. 183

Tables

TABLE 1: SAMPLE METADATA SECTION TABLE 27
TABLE 2: MD.* --TRUE METADATA 38
TABLE 3: ES.* --EXPLOITATION SUPPORT 48
TABLE 4: PF.* --PRODUCT FORMATION 67
TABLE 5: PD.* --PRODUCT DEFINITION 78
TABLE 6: SC.* --SENSOR CONFIGURATION 94
TABLE 7: LD.* --LIDAR DATA 102
TABLE 8: UD.* --USER DEFINED DATA 104
TABLE 9: GEOLOCATION ERROR SOURCES BY PSF AND NON-PSF CATEGORIES. 164

1. Overview

Light Detection and Ranging (lidar) is a technology for the gathering of geospatial data that continues to gain momentum in both the commercial sector as well as in the Intelligence Community and Department of Defense (IC/DoD). One of the major applications of lidar to date has been in the development of High Resolution Elevation (HRE) data to provide 3-D representations of the Earth's surface. However, HRE is only one application of lidar and the number of possible applications continues to grow as the acceptance of the data within the various communities increase.

To date the standardization of the collection, processing, storage, and dissemination of lidar data has been relatively unguided by specifications and standards. There have been a few attempts at standardization such as the development of the LAS file exchange format by the American Society for Photogrammetry and Remote Sensing (ASPRS), the ASPRS developed document Vertical Accuracy Reporting for lidar, and the inclusion of lidar in the Guidelines and Specifications for Flood Hazard Mapping Partners developed by the Federal Emergency Management Agency (FEMA). However, there is a need for additional lidar standards and specifications for lidar data and lidar operations. The development of these standards and specifications are especially important inside of the IC/DoD where there are current efforts to ingest and disseminate lidar data through the National System for Geospatial Intelligence (NSG) using existing tools and infrastructure. The intent of this document is to take the first step of meeting the DoD/IC needs by creating a conceptual model and associated metadata dictionary for lidar metadata.

1.1. Executive Summary of Enterprise Level-2 Lidar Model

This enterprise level-2 model document covers partially processed lidar sensor system data (subject to the Scope defined in section 2 below) and is intended to be the basis for a data definition that can be used as an intermediately processed lidar data standard.

The point of Enterprise level-1 data is to supply a root sensor system data set for the NSG processing, exploitation, and dissemination (PED) architecture. To satisfy the objectives of these intermediate data sets, the level-2 data definition scheme provides for data sets that must

- be processed to points and/or wave form clips (short return histories corresponding a range interval within the scene scattering volume, not available for Geiger Mode collection),
- support interoperability through conformance to unambiguous public definitions
- be a one stop source for all sensor and mission data needed to generate those products for which it has been tailored and that can be generated from a single collection operation's data output.
- provide a source for downstream inheritance of metadata necessary for discovery, dissemination control, and full value exploitation that includes data augmentation due to the creation of new metadata by the processing algorithms applied to precursor products)

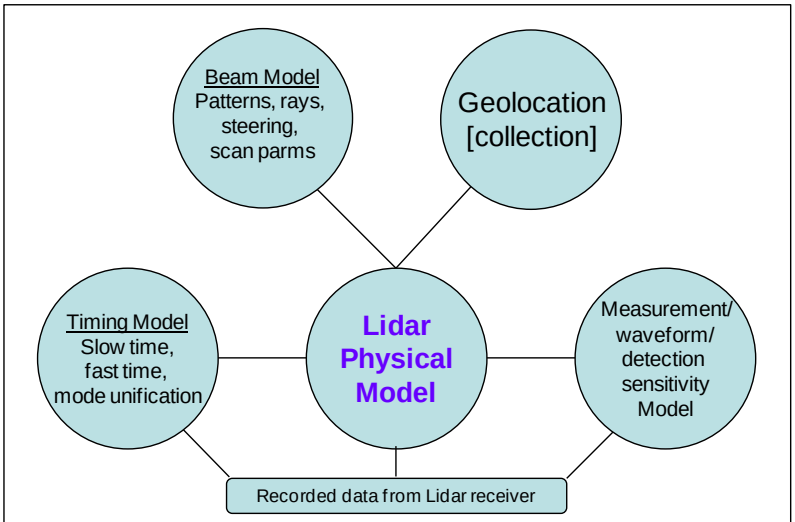
At the core of the data model is the wideband data payload of returns created by analyzing return energy and the precise times at which it arrived. These returns are represented by either or both 3 to 5 D (3-D position, + time of measurement + channel of collection) points and waveform clips (time series of return energy measurement for each sensed look direction, at a particular time and in a particular channel). This is the heart of the model and it is rather simple.

One would then ask, "if the problem is so simple, why are there over 530 items in this lidar level-2 data dictionary?" The answer is that while the core problem of finding approximate positions of scatterers is simple, there is a lot more that we expect from a lidar data chain. Much more physics and geometry is needed to be able to interpret the meaning of point clouds and to assign intensities to points, and understand geolocation accuracy. The data model supports multichannel systems, sophisticated waveforms and scanning capabilities, and downstream processing to enhance quality/utility. Then there are matters such as the need to be able to manage the processing and exploitation of extremely large data sets and support sophisticated library functions on the intermediate data and resultant products.

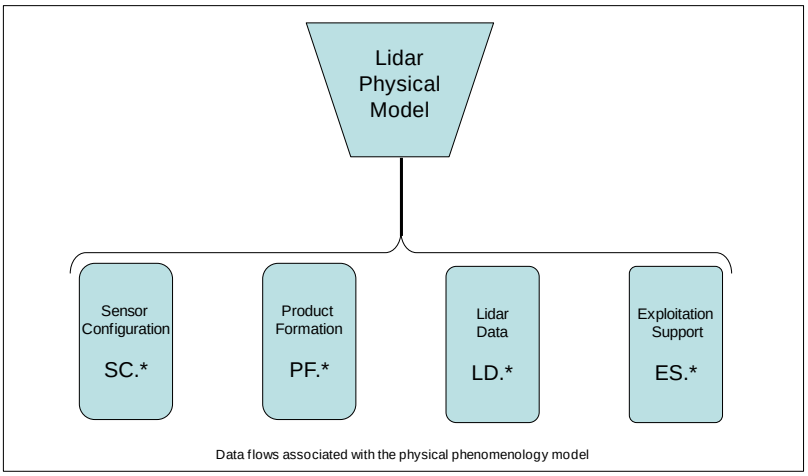
In particular, high performance parallel processing methods are expected to be the staple approach to processing massive quantities of lidar data or to create real time data "waterfall" or 3-D video displays. The break up of the raw data into sections and the saving of results from operations that users only need applied to a limited number of points (which ones are not known at the time of processing), requires a number of metadata items that track the process of processing itself so that low latency, high value products can be generated. Such metadata is defined here for the first time for lidar data. In the past, little regard was shown for the details of when a lidar point was collected, with which other points it has been registered and what corrections may have been determined to be necessary for best quality rendering and geolocation. Consequently the community has been frustrated by typical lidar artifacts such as "cornrows", lack of geolocation error propagation and lack of collection aspect information unless only a single aspect was collected.

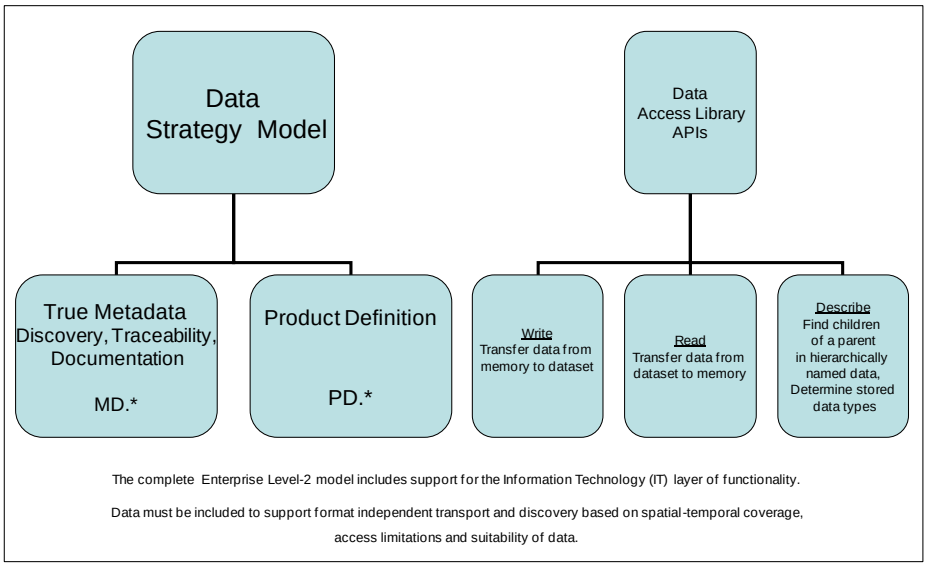
Those capabilities can only be fully realized if we have defined hundreds more data items. While an implementer may chose to cut costs by not supporting various of these capabilities, the data model clearly lays out what the information is for a full capability suite so that system engineering can be done by "Chinese menu" based on selection of a prioritized subset ("profile") of the full data model for population. The bulk of this document is that large set of details where the devil for any real world implementation lives.

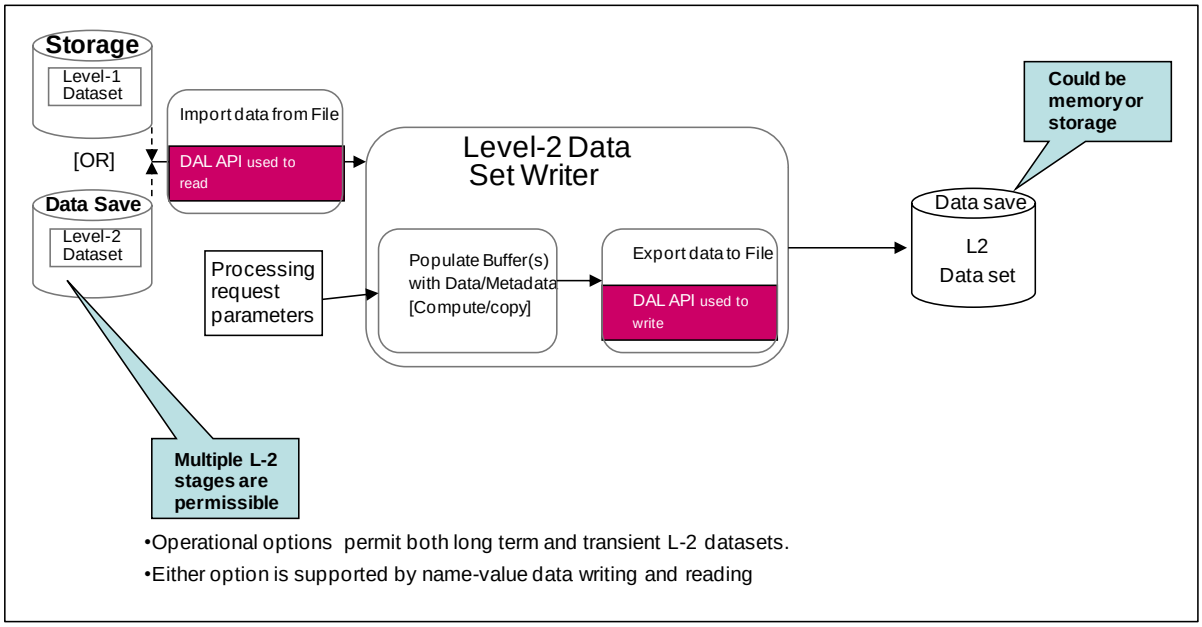
The four diagrams below give a high level overview of how the core physical data is supported by the additional data in the level-2 model.



Enterprise Level-2 Core is the physical model of Lidar receiver measurements. This is supplemented by the sensor properties and the position and orientation of the sensor.


Lidar Physical Model
Sensor Configuration
SC.*
Product Formation
PF.*
Lidar Data
LD.*
Exploitation Support
ES.*
Data flows associated with the physical phenomenology model


Data Strategy Model
Data Access Library APIs
True Metadata Discovery, Traceability, Documentation
MD.*
Product Definition
PD.*
Write
Transfer data from memory to dataset
Read
Transfer data from dataset to memory
Describe
Find children of a parent in hierarchically named data, Determine stored data types
The complete Enterprise Level-2 model includes support for the Information Technology (IT) layer of functionality.
Data must be included to support format independent transport and discovery based on spatial-temporal coverage, access limitations and suitability of data.


Storage
Level-1 Dataset
[OR]
Data Save
Level-2 Dataset
Import data from File
DAL API used to read
Level-2 Data Set Writer
Processing request parameters
Populate Buffer(s) with Data/Metadata [Compute/copy]
Export data to File
DAL API used to write
Data save
L2 Data set
Could be memory or storage
Multiple L-2 stages are permissible
•Operational options permit both long term and transient L-2 datasets.
•Either option is supported by name-value data writing and reading

2. Scope

This document, Conceptual Model and Metadata Dictionary for Enterprise Level 2 Volume Return Products, has been developed by the National Geospatial-Intelligence Agency (NGA) to support the need of the NSG for lidar standards and specifications in support of data discovery, access, processing, exploitation, and exchange. The document provides both a conceptual model for lidar metadata and also a metadata dictionary. The conceptual model provides a formal description of the metadata elements that are required for lidar and a structured arrangement of those elements through the use of conceptual schemas. The metadata dictionary provides the definition for every element. The goal of this document is not to provide examples of encoding lidar metadata into specific file formats. Rather, it is to provide the framework upon which specific usage of lidar data and metadata that conforms to a common understanding of the meaning and use of that data can be performed in the future by all users of compliant data sets and tools. In fact, a data model that enables hiding the format details within an information technology (IT) transport layer (encapsulation) is a key objective in the design of the Conceptual Model and Metadata Dictionary (CMMD).

While we refer to this document as a conceptual model, the objective of this document to be an enabler for data interoperability requires that substantial logical model content be present so that the semantics and pragmatics for each metadata item are sufficiently defined that misinterpretation of the use of data items cannot be ascribed to the treatment in this document, but would rather become cases of non-compliance. The dictionary content extends the document to the logical[1] data model level in order to provide that basis for data interoperability.

This document lays out a format independent data convention and description that permits a modern IT solution to the problem of format specification. The old paradigm was that a data design specific enough to assure interoperability (after testing and validation, of course) would need to include a format specification. In the new paradigm, format is no longer part of a data specification, but is at the discretion of part of the transport layer and is effectively encapsulated so that all software applications that use the compliant data sets can use the same (language dependent) code to populate or extract information from the data files. Such a solution would leverage a Data Access Library (DAL), which would consist of executable modules that provide an application program interface (API) for general writing, reading and query functions to a transport file such that no application that calls the DAL to read, write or query a DAL generated file needs any knowledge whatsoever of the format of the underlying transport file or data stream, including the Endian state of its data. The DAL is not specific to lidar or this CMMD. It can be applied to data from any sensor and CMMDs for other phenomenologies and sensor types can be structured to be able to leverage a DAL.

The key element of the design strategy for a data model that can be format independent is the notion of writing to, querying and extracting from the data set "container" by name. That is, a name or "key" (in the database sense) is used as a unique identifier for a data entity. To get data from a data set of any hidden format supported by the Data Access Library modules, the DAL API is provided a name and the value for the entity going by that name is returned.

---

[1] In so doing, the categories of information identified in SensorML are supplied. While not using metadata to provide the semantics of the metadata as would occur in a SensorML file {except for the CMMD treatment of user defined data}, this CMMD is consistent with SensorML specification of information details.

While the model in this CMMD document stands logically with or without a Data Access Library (DAL), its full power (principally support for fully self defining data sets) comes to bear when a DAL is used to write files that could accommodate population of any data item in the data dictionary in Appendix 20 of this document. See figure 2-1 below.

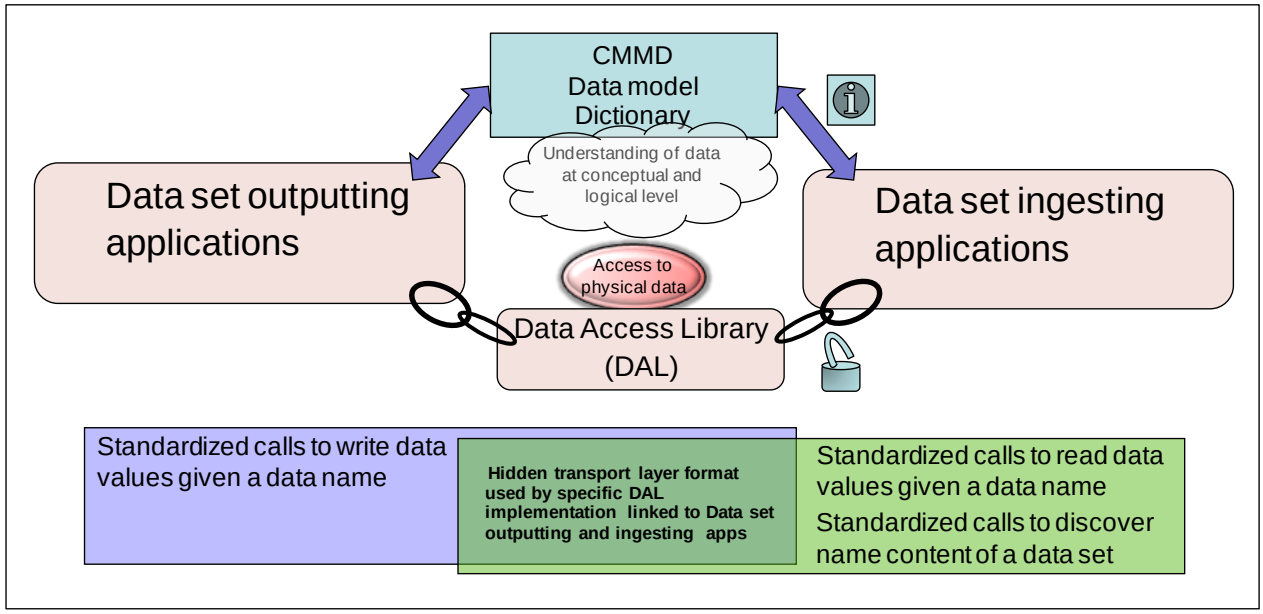


Figure 2- 1: CMMD, DAL and CMMD compliant, DAL managed data sets

The potential instantiations of the DAL may use a variety of formats. Those that permit population of any data item in this CMMD document {such as HDF-5, KLV, SQLite, XML} can provide the full range of capability in the lidar data model. Those formats that use fixed data offsets and positionally inferred meanings such NITF or LAS 1.x would have to have extraordinarily complex, extended or non-compliant implementations to support the full range of data capabilities in this document. The affordability of implementation constraints would likely make such underlying file formats useful only if a limited range of capabilities were required of the data sets written with them.

This document does not govern the DAL; it covers only the compatible model and data dictionary.

This document carries the subtitle, "Conceptual Model and Metadata Dictionary for **Volume Return Products**". Volume Return Product (VRP) is a broad class of lidar data products that emphasize the signal as principally bearing information about scatterers in a 3-D spatial region. Point clouds, surfaces (TIN or DEM style) are the main reductions of this class of data. Preservation of certain focal plane element time histories ('waveforms') within subsets of the receiver recorded range interval (often referred to as a "range gate"[2]) is also supported in this conceptual model. Lidars are also capable of

[2] Range gate is also a term that is used to indicate the time intervals corresponding to return activity from distinct thin shells of scene volume that are *resolved* in range. So we avoid using "range gate" as the term for the subset of the interval associated with each pulse in which data are recorded for all ranges that the instrument has been set to be sensitive to. This is handy for discussing Geiger mode, which might be better said to employ a large number of range gates for each pulse return. The time between pulses that a LIDAR receiver is sensitive and

providing coherent time series such as might be used to monitor sub- and audio- frequency modulations of remote scattering surfaces, Frequency shift data, RF modulated waveform returns, aerosols and temperature etc. These alternative sensor objectives may use elements of the current VRP data model, but the current document is focused on an early set of GEOINT priorities and is biased toward lidar receiver outputs that do not have a phase component and are the result of time of flight measurements of photonic echoes. So, we cover VRP, but not all lidar phenomenologies. That being said, point clouds measured from laser scanners that do not use time of arrival to determine range can still have their data processed into enterprise level-2 and level-3 point clouds which can be fully conveyed via this conceptual lidar data model. Also, a temporal sequence of 3-D point cloud scenes, effectively a 3-D movie is representable within this metadata and data sorting framework.

An important issue of scope for this document involves the distinction between "lidar data" and "lidar derived" data.
For purposes of this discussion, lidar data

- includes measurements directly made by a lidar sensor (time domain data, "waveforms")
- includes points that are times rescaled to ranges and the directions in which the lidar sensor took the time-of-photon-flight measurements (polar coordinates)
- includes sensor frame or earth frame ranges and directions and their transforms to Cartesian coordinate points
- includes measurements of the wavelength, polarization, intensity of photons scattered back to the sensor.
- Includes the output of a filter that represents some combination of lidar data of the above types-principally to include coincidence processed lidar points as lidar data.
- level-1 and level-2 data due to lidar collection are, by definition, limited to being "lidar data".

On the other hand, "lidar *derived* data"

- Is generic data that could have been made by processing data from a non-lidar sensor but happens to be due to lidar collection
  - DEMs and related products (such as HRE, HRTE, DTED, DTMs) that give a height for each horizontal position on a grid.
  - Contour maps
  - TINs, Scene models
  - Parametric extractions of feature geometric properties (e.g., vertical object data)
  - Feature recognition codes, object classifications
- Level-4 data due to lidar collection may be either "lidar data" or "lidar derived data"

The scope of the enterprise level-2 lidar CMMD applies only to lidar data, not lidar derived data[3]. Lidar derived data definitions are to be governed by documents specifically focused on their generic content and possibly, format.

---

may have individual gates correspond to range resolution cells is called a "receive window" in this document. The spatial range interval associated with complete returns for points collected within a receive window is called the "range swath".

[3] Derived data would be Level-4 by the definition in section 5.1. The lidar CMMD Level-4 document will only address lidar data. Other documents (for example, the HRTe spec) govern products that may be derived from lidar data that no longer contain lidar wideband data per se.

For implementation guidance, a series of level specific lidar CMMD documents has been generated. These documents permit implementers and users to focus on the functionality of the product levels with which they are responsible. This level-2 document has the largest content among the documents in the set as it retains almost all metadata inherited from enterprise level-1 as well as adding new metadata from the transformation of the the time domain data at level-1 to lidar points. Many items needed for processing of raw data are still needed in lidar Enterprise level-2 data sets owing to their mission as intermediate products capable of conversion into a wide range of downstream products or providing support to the process of evaluation of confidence metrics for the extraction of quantitative scene characterization parameters. Furthermore, at enterprise level-2, a mix of any of several various stages of point cloud formation with wave form clips is supported for maximal down stream exploitation procedures support. Thus, many parameters that also occur at level-3 will be introduced in this document. The level-1, 3 and 4 documents afford a fair amount of reduction in discussion topics.

3. Conformance

3.1. External Related Data Standards Documents

This document was developed with the intent of being in conformance with the National System for Geospatial Intelligence Metadata Foundation (NMF) document developed by the GWG Metadata Focus Group (MFG) and other pertinent metadata documentation. These external documents are undergoing periodic revision and the versions in effect when this CMMD was generated may not have taken into consideration issues that became apparent in the constructin of the lidar data model. Where a need to add elements or modify elements conceptually related to items addressed in external standards documentation arises, this CMMD presents the new content and it is recommended that actions be initiated to raise the issue to the relevant GWG subgroup to upgrade the collective documentation to a consensus with a more sound engineering solution. It is expected that as later versions of this document are released, a greater level of reference and linkage to the evolving NMF will take place, with many metadata items first appearing here eventually migrating to a common document at a higher level.

3.2. Registry Recomendations

This document defines a number of metadata items whose values best support enterprise interoperability when they are commonly understood. The basis for common understanding of the bulk of these items is provided merely by the definitions within this CMMD. However, some values, which are either numerical codes or character strings, ought eventually to be listed in registries to facilitate implementation within the NSG enterprise. Generally, constraints for interim values for such data items are proposed in this CMMD to bridge the interval between the publication of this document and the eventual establishment of registries for the data item values for the NSG. The individual appendix 20 entries for the items in the following table suggest that their values ultimately be controlled via a registry.

CMMD Metadata Item	Registry Recommendation
MD.Objective.Tasker	[official spellings of organization designators]
MD.Sensor.ID(j)	[official sensor identification/designation list]

SC.Artifacts.Descr(j)	[official list of sensor origin data artifacts]
PF.Adjustmts.Artifacts(j)	[product artifact name/descriptor registry]
PF.Channlzn.Hist.Comb.Alg.*m*(j)	[algorithm names/descriptors]
PF.Recon.RootDS(j)	[list of designations for NSG domain data product archives]

In addition to registration of values for metadata items, a registry of logically equivalent metadata names (also called tags or keys) would facilitate interoperable implementation of processing applications for product generation and exploitation. Adhering to the spellings of tags as provided in this CMMD document can provide interoperability. But this may not be so easily accomplished, especially when some items defined with dot delimited hierarchical tags in this CMMD are to be ultimately populated in conformity with normative references that may, for example, use camelcase style spelling structures. It is explicitly not the objective or expectation that every reference to a metadata item defined in this CMMD in a compliant dataset, processing algorithm, or tool use only the spellings of tags as listed in this document or the lidar CMMD series of documents. These names are in many cases rather long and it is understood that abbreviated references will naturally arise in various implementations. So long as all logical equivalents to CMMD tags are available in a registry of replacement spellings, these alternative names may be used in transferable files without violating reasonable CMMD compliance criteria. Certainly, substitutions within software that are only referenced internally are of no threat to interoperability and may or may not conform to registered aliases for CMMD metadata item names without the need for compliance testing or verification. Use of aliases could complicate DAL implementations which fully support wild card referencing of the hierarchically structured data tags, but importing key content from such an alias registry would permit interfaces to work as if the CMMD spellings were always in place even though different software actually used different entries from the logical name replacement table constituting the data in the alias registry. While informal alias tables might be used among early adopters, this CMMD recommends that the metadata tag registry be instituted as soon as possible in conjunction with operational fielding of CMMD compliant processing capabilities. The standup of the registry for logical name replacements would be a natural component of a data standard compliance verification regime.

4. References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies. Compliance with relevent sections of Normative references is required [4]for compliance with this document. Informative references have no compliance requriements associated with them.

4.1. Normative references

- Department of Defense Discovery Metadata Specification (DDMS), version 1.4.1, 10 August 2007(https://metadata.dod.mil/mdr/irs/DDMS/)

[4] We must recognize that the normative references may be undergoing a process of maturation and adherence to flaws in current versions of a normative document is not required. We should anticipate that later revisions will correct such shortcomings. This document strives to comply with them except where they are in need of revision for practical reasons.

- High Resolution Elevation Product Specification (Draft), Version 0.9, September 2008
- ISO 19123:2005 Geographic information - Schema for coverage geometry and functions
- ISO 19115:2003 Geographic information – Metadata
- ISO 19115-2:2008 Geographic information - Metadata: Extensions for imagery and gridded data
- ISO 19107:2003 Geographic information – Spatial schema
- MIL-STD-2500C National Imagery Transmission Format Version 2.1
- National System for Geospatial Intelligence Metadata Foundation (NMF) (Final) Version 1.0
- STANAG 2211 (Edition 6): GEODETIC DATUMS, PROJECTIONS, GRIDS AND GRID REFERENCES
- STANAG 4545, NATO Secondary Imagery Format (NSIF)
- STANAG 7023, Digital Geographic Information Exchange Standard (DIGEST)
- Recommendation for Space Data System Standards BLUE BOOK TIME CODE FORMATS RECOMMENDED STANDARD CCSDS 301.0-B-4 November 2010
- XML Data Encoding Specification for Information Security Markings, Version 7 2011-08-09

The above list of normative references identify documents that themselves will be subject to revision and re-publication. The GWG is adopting a process for maintaining harmonization into the future among documents such as this CMMD, which govern the generation of datasets that will need to conform with these other documents as well. A case in point regards security markings. The plan is for this information to be eventually removed from this CMMD and for implementers to use what will be registered names when populating security marking fields in name-value supporting data containers in the manner described in this CMMD.

4.2. Informative references

- High Resolution Elevation Product Specification (Draft), Version 0.9, September 2008
- Data Access Library API Design Description, NGA/IID internal document (2009)
- USGS lidar Guidelines and Base Specification Document also referenced by FEMA. http://lidar.cr.usgs.gov/USGS-NGP%20Lidar%20Guidelines%20and%20Base%20Specification%20v13(ILMF).pdf

5. Terms, Definitions and Conventions

5.1. Terms and definitions

ambiguity

Active remote sensing systems have the ability to portray responses from systematically[5] different locations as being due to a single source position. An ambiguity is one of the incorrectly positioned

[5] The word,"systematically", is used to exclude *random* position errors due to uncertainity in the sensing system's precise state.

reports of collected energy. A common source of such ambiguous response is the use of modes that have many pulses in transit at a time. The number of of pulses on the roundtrip path is called the *ambiguity* number.

application
v. manipulation and processing of data in support of user requirements [ISO 19101]
n. A software program that performs a specific set of manipulations or transforms of input data to create an output set of data, some or all of which would be displayed to a user or passed as data to another application. Applications are distinct from system software and "middleware", which drive hardware functions at an infrastructure level. Signal processing programs are examples of applications.

application schema
conceptual schema for data required by one or more applications [ISO 19101]

azimuth (collection azimuth)
the angle measured counterclockwise from north in the north-east-down frame of the intersection of the plane of incidence[6] and the earth tangent plane at a point of observation on the tangent plane

channel
In this lidar CMMD document, channel is used only to refer to lidar channels, not other more general channels such as communications channels, data device recording channels, etc. Thus, the implied adjective, "lidar", should be understood in all contexts internal to the lidar CMMD documentation series. A lidar data channel refers to a unique phenomenological recorded data stream set. For more details regarding the significance of the term, "phenomenological" in this definition, see Appendix section 10.1.3.1.

[6] This is the conventional physics defined plane of incidence, specified by the normal to the earth's tangent plane at the observed point on the earth and the line of sight.

channelization
In this lidar CMMD document, a channelization is a choice of physical phenomenology assigned to a set of channels. A multichannel sensor collects an original channelization based on its design and the channelization consists of what distinct values of electromagnetic field parameters are assigned to the wideband channels. {Choices include the spectral band, polarization, look direction, modulation}. Processing can introduce new channelizations by combining channel data per some algorithm. Such combinations may or may not reduce the number of channels in the new channelization compared to the earlier channelization.

class
description of a set of objects that share the same attributes, operations, methods, relationships, and semantics [adapted from ISO/IEC 19501]

Collection operation, Collect (*n*)
Collection operation refers to the action of a sensor system in the original obtaining and recording of remote sensing data, and a **Collect** is the body of data so collected in a single collection operation, i.e., an interval of continuous operation of the sensor employing a single set of sensor configuration parameters.

conceptual model
model that defines the concepts of a universe of discourse [ISO 19101]

conceptual schema
formal description of a conceptual model [ISO 19101]

conceptual specification
declarative description of conceptual schema

conformance
fulfillment of specified requirements [ISO 19105]

data set
identifiable collection of data [ISO 19115] This document frequently uses the term “current data set”, which refers to the instantiation of the product data in which a metadata item under discussion is being populated, or from which it is being read. For a generic remote sensor, a very large number of data products are eventually made. Many of those use common metadata entities with specific values populated into individual data sets. The “current data set” reference intends to indicate the particular data set under discussion and not other similar or related data sets that would have a different data set name.

data type
specification of a value domain with operations allowed on values in this domain [ISO 19103]

EXAMPLES An Integer, a Real, a Boolean, a CharacterString, and a Date.

NOTE Data types include primitive predefined types and user-definable types.

dwell
A dwell is a contiguous interval of time during which data for a single position in space is collected. Thus in the case of a lidar, a dwell refers to a consecutive string of IPPs for a pulsed lidar, that is used to interrogate a scene volume element. Thus, there is a potential to collect return photons for each pulse in a dwell for a single 3-D scene position. Absence of such returns indicates lack of scattering material or extinction of the beam, e.g. by obscuration.

frame
A frame of lidar data is the data collected in a single IPP.

geospatial
refers to either an implicit or explicit reference to a location relative to the earth.

geospatial information
information that identifies the geographic location and characteristics of natural or constructed features and boundaries on the earth, including statistical data; information derived from, among other things, remote sensing, mapping, and surveying technologies; and mapping, charting, and geodetic data, including "geodetic products." [Title 10 US Code 467]

geospatial intelligence
the exploitation and analysis of imagery and geospatial information to describe, assess, and visually depict features and geographically referenced activities on the earth
NOTE Consists of imagery, imagery intelligence, and geospatial information

geospatial intelligence standard
documented agreement containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics to ensure that materials, products, processes, or services are fit for the analysis and visual representation of physical features and geographically referenced activities.

implementation
realization of a specification. In the context of the ISO geographic information standards, this includes specifications of geographic information services and data sets [ISO 19105]
As used in this document, an implementation is an operating or operable realization of a design for hardware or software.

Interpulse Period (IPP)
An interpulse period is the temporal interval between consecutive pulse transmission times and beginning with a pulse transmission event. See 10.1.1 discussion. Each interpulse period in a lidar collection operation has an index, being an integer from the consecutive sequence 0,1, 2, 3…N, where N is the total number of pulse transmission commands minus 1. During an interpulse period, the data for creating waveforms and lidar points is recorded. So each natural lidar point has a unique interpulse period with which it is associated. The reciprocal of the IPP is the Pulse Repetition Frequency (PRF).

Latitude
Latitude is taken as the geodetic latitude in the WGS84 system. It is the angle between the ellipsoid normal and the equatorial plane of the earth. It is measured in decimal degrees with zero degrees at the equator, positive latitudes in the northern hemisphere and negative latitudes in the southern hemisphere. CMMD compliant values are signed as described here, with no north or south designation being used.

Level [also, enterprise product level]
Product levels are referred to as "level-0", "level-1", "level-2" etc. These are references to Enterprise functionality and purpose for data sets and the data within them. Level-0 is not intended for PED use and due to its 'vendor internal' nature need not be interoperable with the rest of the enterprise. The levels 1 through 5 should be implemented in an enterprise interoperable manner and conform to standardized data definitions.

Level-0

This level is sensor system design unique data whose use requires deep understanding of the developer's conventions, codes, internal models and data stream structure. It is customarily not designed to meet any external interfaces, is not documented for interoperable use, and is viewed as internal data from the remote sensor hardware vendors' point of view. It includes the encoded forms of wide band and narrow band data, commanding data, system status data.

Level-1

This level is raw data or data losslessly invertible to raw data that is expressed in terms of publicly defined metadata and wideband data entities based in physics and geometry. It is complete and unfiltered sensor signal output and includes all supporting data tables necessary to convert measurements to engineering units, including calibrations and known transfer functions. The level-1 data includes only the relevant conversion functions and does not require the "user" to select the appropriate data from a library. All necessary sensor/platform models have been exercised and their physical and geometric outputs included in the level-1 data. level-1 data is processor ready as the method of application of its data items is defined in the data dictionary. *Aggregated* level-1 data is also a grandfather data set that supports inheritance of data to higher levels of processing, dissemination control and exploitation. An example of data that might be aggregated in a level-1 file is error covariance support data, used in downstream error propagation.

Level-2

This level is substantially processed data that is in an intermediate state that is generally not ready for human visual interpretation in its entirety. It may, in certain cases, be analyzed via numerically based tools, but exists primarily as a common processing tap point for divergent subsequent processing threads that lead to primary exploitation products at level-3. A lidar derived data set that has both XYZ points and waveform clips is level-2.

Level-3

This level consists of primary exploitation ready products engineered for human visual interpretation. Level-3 products are derived from single raw data sets (single collection operation of a single sensor).

Level-4

This level consists of products whose wide band data are derived from, or represent the fusion of, multiple level-3 data sets[7]. This includes products that combine multiple collects from a single sensor type as was well as level-3 data derived from different sensing system types.

One might ask if a level-4 data set combines lidar with, say, an EO data set, does the lidar CMMD govern the data set or does the EO CMMD govern the data set? The answer is that when level-4 documentation is broken out of the lidar CMMD in a later release, it will be apparent that it does not matter. The DAL transport independent approach permits combining of metadata from any of the CMMDs into a single product. The organization of the data may be chosen to correspond to the lidar point cloud and leverage the tiling constructs of this document when it is advantageous.

Level-5

This level consists of more than just data. Judgments and interpretations have been imposed via methods such as filtering, annotation of analyst observations etc. Data that has been subjected to and carries the results of automatic or manual target recognition, has highlighted, color designated, or zoomed-in sub-regions, carries notations of extracted information such as object identification codes, velocity (speed + direction), prioritizations for analyst attention, etc are level 5. Models derived from level-3 and level-4 data are level-5.

Lidar Point

A lidar point (also, "natural lidar point") is a spatio-temporal location specified by a selection of a ray and a range on that ray measured from the ray origin in the sensor. It is typically represented in XYZ ground space coordinates and the (slow) time of the ray. For this discussion, a ray is the photon path associated with a specific sensing element look direction and a specific time. For moving sensors, the specialization to a specific time is equivalent to selecting the sensor position and general look direction. For a single channel lidar (thus a single focal plane sensing array of sensing elements), a point is a 4-D entity. For a sensor that simultaneously collects multiple channels, the choice of which channel has the element on which a ray originates constitutes a 5th parameter for a unique specification of a lidar point.

Under this definition, a lidar point corresponds to the measurement of received photon flux at a specific (slow) time from a specific range in a specific direction and in the spectral and polarization pass band associated with the specific sensing channel from which the measured data was recorded. Returns from the same XYZ ground space position at two different times or as measured in different sensing channels at the same (slow) time would constitute separate and distinct level-2 (5-D) lidar points. At level-3, the channels do not act as a point dimension, so level-3 points are, at most 4-D. See also "Synthetic lidar Point".

Lidar Waveform Clip

A lidar waveform clip is an interval of fast-time return intensity measurement. A full waveform clip would span the duration of the complete receive window for an inter-pulse

[7] A level-2 or -3 product may have its metadata accuracy improved by updating from a level-4 product. Until the wideband portion of the product has multiple sources, the product is not considered Level-4.

period. Typically, short sub-intervals might be saved in level-2 products, while level-1 products would have only full waveform clips.

Longitude
Longitude is taken as the geodetic longitude in the WGS84 system. The longititude of a point on the ellipsoid is the angle between the plane containing the rotational axis of the earth and that point and the plane containing the prime Meridian[8] and the rotational axis of the earth. Longitude is measured in decimal degrees with zero degrees at the prime meridian, positive longitudes east of the prime meridian up to 180 degrees and negative longitudes west of the prime meridian to -180 degrees. CMMD compliant values are signed as described here, with no east or west designation being used.

metadata
data about data [ISO 19115]

metadata element
discrete unit of metadata [ISO 19115]

NOTE 1 Metadata elements are unique within a metadata entity.

NOTE 2 Equivalent to an attribute in UML terminology.

metadata entity
set of metadata elements describing the same aspect of data [ISO 19115]

NOTE 1 May contain one or more metadata entities.

NOTE 2 Equivalent to a class in UML terminology.

model
abstraction of some aspects of reality [ISO 19109]

nadir angle
The nadir angle is the (cone) angle from the vertical to the line of sight. In order that geometric relations for viewing of the target work independently of the range between the target scene and the sensor, we define the nadir angle in terms of a (gravitational) vertical (i.e., plumbline) direction *at the location of the target scene* being sensed. For near range viewing from altitudes above the nominal horizontal plane at the target, the sensor referenced vertical direction will be negligibly different from the target based vertical direction. While gravitationally referenced versus ellipsoid referenced choices of vertical can be critical for the utility of a computed point cloud, the nadir angle participates in computing characteristic properties of the point cloud (such as visibility of vertical surfaces and projections of sampling density to the horizontal plane) that are not impacted by the gravitational deflections of vertical, which are generally less than an arc minute. Nadir angles are more nominal values and the approximation of quantities derived from the nadir angle is not degraded by ellipsoid to geoid vertical reference differences. Likewise, very sensitive angle based calculations such as ray bending effects will be based

[8] IERS Reference Meridian (IRM), also called the International Reference Meridian, which is the reference meridian (Prime Meridian, 0° longitude) of the Global Positioning System operated by the United States Department of Defense, and is the reference meridian in WGS84

on coordinates and directions that are more precisely defined than nadir angle, so the nominal nature and target referencing of this definition do not compromise such high precision calculations. Typically,airborne sensors may substitute sensor referenced nadir angles values without violating the quality constraint on nadir angle values.

For airborne sensors, typically the value of nadir angle ranges from zero (directly down along the gravitational vertical) to some value less than 90 degrees indicating dominantly down looking directions. For ground based sensor that might be aimed in a dominantly horizontal direction, the nadir angle would range from something like 70 or so degrees to 120 or so degrees.

Narrowband
This refers to data within a data set that is carried by entities whose byte volume is small, i.e., a small fraction of that due to the direct sensor content of a data set. Distinct from wideband.

National System for Geospatial intelligence
the combination of technology, policies, capabilities, doctrine, activities, people, data, and communities necessary to produce geospatial intelligence in an integrated multi-intelligence, multi-security domain environment

PED
Processing **E**xploitation **D**issemination used in the sense of a system architecture for creation, distribution and use of GEOINT data sets.

point cloud
A point cloud is a sparsely filled 3 (or greater)-D array. It is usually not stored as gridded data. In the lidar case, a point cloud is built up from a list of (x,y,z)-tuples representing the positions of lidar measurements (that is, lidar points or synthetic lidar points) that have each been reduced to a representation of a data sample at a geometric point. Such a reduction is done via some model of the scene, the lidar sensor and applied data processing algorithms. The geometric points have some inherent positional uncertainty that results in them only being constrained to lie in some error ellipsoid to some level of probability. The spatial resolution of the sensing system and processing are contributors to that uncertainty. The point cloud entry is generally to be considered at the center of such an error ellipsoid. In the terms used in this CMMD, we may attach additional information to a point cloud by associating values such as intensity or return number with each XYZ listed in the point cloud.

profile
set of one or more base standards or subsets of base standards, and, where applicable, the identification of chosen clauses, classes, options and parameters of those base standards, that are necessary for accomplishing a particular function [ISO/IEC TR 10000-1]

NOTE For profiles that are created as a combination of complete base standards, conformance to a profile is conformance to all of the base standards of which it is composed.

As used in this document, the term “profile” is used to indicate a subset of content or a specific population strategy for data described by a more general document. Implementations of data sets generally use a profile of the data defined in its base document.

range swath
The spatial range interval associated with complete returns for points collected within a receive window is called the "range swath". This is to be calculated assuming no range ambiguous returns are present. Thus, a range swath applies to one ambiguity and does not add up the range return intervals from the ambiguity number of ambiguous intervals.

remote sensing
Collection of information from a target by observation from a distance large compared to the target size.

resource
asset or means that fulfills a requirement [ISO 19115] [9]

return
A lidar return is the sensor response in a single channel to photon energy in the recording interval associated with a single Interpulse Period that corresponds to scattering from a distinct (small) range interval along a distinctly recorded ray. In a Geiger mode system, there can be a return for each quench period spanned by the recording interval within an IPP. But if no photons are sensed in a quench interval, there is no return. In linear mode, a return is usually associated with a peak in the recorded photo-current. However, any interval of recorded photocurrent can be considered to be *a* return if the photocurrent is above the system noise floor and there are no range ambiguities. For pulse compression lidars, the returns from narrow ranges may be spread over a time interval long compared to the light transit time across the reflecting range. Until the pulse is compressed, the returns from objects at different ranges along a ray will not be resolvable as distinct responses. Note that the term, "return", is defined on an IPP and channel basis. Consequently, a return could be range ambiguous, meaning that distinct range intervals may share time intervals within the recording interval of one IPP. That is, echoes due to scattering from multiple distinct range intervals could be superposed in the receiver. Also, a return expected from a particular target range could be actually due to a different transmit pulse than intended and consequently from a different range than expected. This is a consequence of operating with PRFs high enough that multiple pulses are on the roundtrip ray path at a single instant. In general, for a single pulse and a single ray, a lidar channel may record multiple returns. [If range ambiguities can be ignored, earlier returns are from closer to the sensor.] Return counts for a pulse are given separately for each channel.

In the case of channels that have been derived by combination of collected channels, returns may still be identifiable, particularly in the case of time domain wave clips. When the channel combination corrupts the meaning of return as a logical extension of the above discussion for a collected channel, return numbers should not be populated as they can not be well defined. The processor, when faced with only a small fraction of points having undefined return value when creating a synthetic point set, has the tile-level choice to edit out the problem points or not populate the return code for any point.

[9] Most references later in the document treat data sets as resources. This definition is not consistent with the more customary use of the term "asset" as hardware including sensors, platforms and comms. Resource, as used by the NMF and its source ISO documents, refers to a published document or data set. It is an aggregated body of information, whether or not there exists a "requirement" for it.

specification
declarative description of what something is or does [adapted from ISO/IEC 19501]

stage [also, processing stage]

A processing stage indicates an intermediate processing step internal to a chain in the generation of an enterprise level-2 or level-3 product. During development phases for a system, there may be exchanges of data at a particular processing stage. For lidar, the stages (previously known as levels) are defined as:

Stage-0
This stage is a new name for what was previously referred to as simply "level-0" data.

Stage-1
This stage is for noisy XYZ point cloud data and is a level-2 product.

Stage-2a
This stage is for data de-noised by coincidence processing. It is a level-2 XYZ product as well.

Stage-2b
This stage is an old designation for scan to scan registered swaths. When exclusively point cloud data, this is a level-3 point cloud.

Stage-3
This stage is level-4 and uses separate collects with angular diversity to improve geolocation accuracy

Synthetic lidar Point
A synthetic lidar point is derived from multiple lidar points by a process of filtering. The filter assigns a value to XYZ and slow time (and optionally intensity, spectral and polarization parameters) that is derived from the input lidar points and the filter weighting algorithm. When multiple slow times are involved, the filter output time represents a temporal smearing of returns. While the time is represented as a single number (in seconds or pulse number units), proper full specification of synthetic lidar points requires a time "spread" data item. When the filter inputs multiple spectral or polarization state values, those too have spreads that are part of a proper full specification of a synthetic point. The projection of lidar points and synthetic lidar points into 3-D ground space leaves only the XYZ values. Often only these projections of the 4/5-D lidar or synthetic lidar points are carried in a data set or displayed by a visualization tool.

uncertainty
As used in this document, uncertainty is a *statistic* of the distribution of measurement error for some quantity such as a distance or a time interval. Typically, uncertainty would be quantified as the standard deviation of the distribution of measurements or a percentile of error magnitude such as the 95% level of the cumulative error distribution.

unit of measure (also, **units)**
reference quantity chosen from a unit equivalence group [adapted from ISO 31-0, 2.1]

EXAMPLE The unit of measure "metre" from the category "length".

All units for physical quantities used in this CMMD are SI units, previously called MKS for meter, kilogram, second.

Vertical
Commonly *local* vertical, this term refers to the direction of the WGS-84 ellipsoid normal through a point at which the local vertical direction is being supplied. This is in distinction with the term "plumbline", which is in the direction of the gravitational potential gradient. Local plumbline directions close to the earth's surface are given by the outward normal to the WGS-84 EGM geoid (of relevant epoch) and are called ES.Align.LocG_Vert(j_tile,j) in this CMMD.
At the surface of the Geoid, the plumbline direction is directly defined. At a position above the geoid, the issue is complicated by the fact that plumblines are curved. Thus, it is much easier to define/compute local vertical directions for objects at a significant altitude above the geoid or ellipsoid in terms of the ellipsoid normal rather than the local gravitational acceleration vector. This also eliminates the need to track whether the earth's rotation is, or is not, a component of determining the vertical direction.

Wideband
This refers to data within a data set that is carried by entities whose byte volume is very large, i.e., comparable to that of the direct sensor content of a data set. Distinct from narrowband.

5.2. Conventions

Hierarchical tag name structures
This CMMD employs the strategy of naming the data entities with dot delimited hierarchically ordered character strings. Each metadata element has a name of the form
Class.Property.TrailingNameDetail
The *Class* substring in the tag for any lidar CMMD data item will be one of the following two letter strings

MD --- True Metadata items associated with discovery, handling and pedigree
ES --- Exploitation Support
PF --- Product Formation
PD --- Product Definition
SC --- Sensor Configuration
LD --- Lidar Data
UD --- User Defined wideband

See section 8.2 and section 9 to find illustrations of the tree structure of the data hierarchy. The *Property* substrings are indicated in the diagrams of sections 9.1 through 9.8. Note that sorting (mostly alphabetically) the data tags when they are structured this way creates a useful overview of the data model. This is most readily seen by examining the table of contents for Appendix 20.

Instantiation tracking via evaluated tag name substrings
Note that in this document, the use of italics for part of a data tag name indicates that the italicized substring is to be evaluated and populated names in the name-value file will have substituted strings where the italics appear in the metadata dictionary. Principally, the strings, "*TileNum*" and "*Ia*" may appear in the dictionary but will be replaced by other strings when compliant data sets are actually created. This supports the unique association of a tag to a data item so that name-value

retrieval is unambiguous. It also eliminates the need to create table entries for the multiplicity of data items within this document.

Angles

All angles are defined as counter clockwise increasing when viewed along the axis of rotation toward the coordinate origin of interest. Rotations about axes through the sensor are viewed from target toward sensor. Rotations about axes through points on the earth are viewed from large radius to small radius toward the point with respect to a right handed system (e.g. North-East-Down NED)

Array and Beam

The lidar illuminator is generally a laser and the information about the illuminator, also referred to as "transmitter", is given by terms relating to the "beam". The sensing part of a lidar is on a focal plane and the individual sensing elements (there may be one or many) are on an array. The term, "array" will be used for the receiver geometry discussions and the term, "beam" will be used for transmitter geometry discussions. Emerging designs may involve the use of optical fiber or other components in such a way that the physical location of transmitter and receiver elements may be displaced from their effective[34] positions. We will be specifiying the effective positions and directions for beam and array when describing lidar geometry.

Coordinate Frames

This lidar Conceptual Model and Metadata Dictionary defines point clouds exclusively in local Cartesian frames. In particular, the vertical direction, Z, is the same at every point in a single tile's point cloud. When northing and easting are used for horizontal positions, they refer only to the UTM position of Cartesian frame origins, not to lidar point position coordinates. While sensor based frames and cylindrical or spherical coordinates are permitted, they apply only to Cartesian frames, with a single value of basis vector direction for each basis vector in any tile and all basis vectors orthogonal within a single tile. Consequently the Euclidean metric of Pythagoras' theorem applies to find the distance between any two points in a single tile. It may be necessary to convert the coordinates of points in separate tiles to ECEF coordinates before evaluating their separation via the Pythagorean rule.

Horizontal Coverage

Per the definition of MD.Coverage.Bounds.Z(j), unless other wise specified, CMMD compliant data sets may only claim horizontal coverage where at least 30 meters of collected vertical coverage is present. This prevents the misleading claiming of coverage in regions near boundaries where tall objects may be truncated due to oblique collection geometry. MD.Coverage.Bounds.Z(3) may be used to indicate a value other than the 30 meter default.

"Nominal" and "Representative" Values

In a context where the term "representative" is used in this CMMD to apply to a quantity subject to a range of values in an ensemble of measurements, we mean a value that is (potentially interpolated to be) at the mean plus one standard deviation, which for normally distributed data is roughly the 84th percentile.

In a context where the term "nominal" is used in this CMMD to apply to a quantity subject to a range of values in an ensemble of measurements, we mean a representative value that is determined from a *model* of the data generation process and does not represent a statistic rigorously extracted

from examination of all of the data in the data set or tile as contextually appropriate. A nominal value is a "characterization" rather a than a pure statistic. When we say we are "characterizing" the data, we are providing a *nominal* value.

6. Data Types

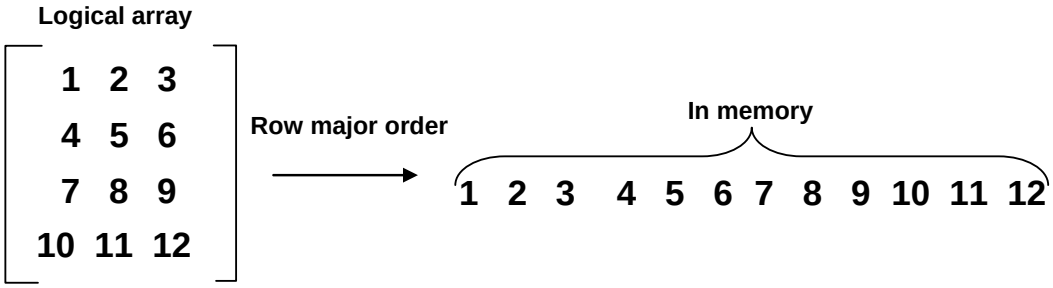
This section provides definitions for data types used in this document. The document follows the data types identified in the NMF Conceptual Schema and uses data types falling into two classes; abstract data types and NMF Conceptual Schema data types. Both types are described in detail below.

6.1. Abstract Data Types

The abstract data types are described below and are based on ISO/TS 19103.

Arrays

For data which are stored as a contiguous array such as **Matrix**(j,k), this *document treats the storage as row major*. This means that ordering indices by the range of the index from high to low results in more contiguous storage for later indices. Arrays appearing in this document may have from 1 to 4 indices.



Any application written in a column major language should reverse the index order where needed to avoid significant run time penalties due to strided memory accesses.

Boolean

A Boolean data item carries a representation of the value “TRUE” or the value “FALSE”

CharacterString

A sequence of 1-byte codes for symbols meant to display as human readable text
For compliance with this document, populated character strings will all be from the Basic Multilingual Plane character set and use UTF-8 encoding. Typically this will be referred to internally as BCS (for Basic Character Set), using the same nomenclature as the NITF documentation.

Codelist

~~value domain including a code for each permissible value~~

~~NOTE It is an open Enumeration that may be extended during system runtime by adding additional named literal values.~~

Note: All codes in this CMMD are integers and will be documented to be of type UInt8 or Int16

Date

Note: All dates in this CMMD are Truncated Julian Day (TJD) integers and will be

documented to be of type UInt16. When the wrap around date comes in 2147, the base date for TJD will be shifted and all data sets generated on or after the TJD shift date will use the new base. Thus, no additional epoch metadata item is needed to keep dating via TJD continuous and unambiguous. However, to keep time differencing accurate when late dates are after the wrap and early dates are before the wrap, either the difference algorithms will need to account for the date that the TJD info was populated, or the new base date adopted in 2147 should be a well past date, such as 2100, so that the likelihood that dates on lidar data sets that are to be checked for their temporal separations would span the wrap date is negligible or controllable to be zero.

Enumeration

data type whose instances form a finite list of distinguished values

NOTE 1 The extension of an Enumeration type implies a schema modification.

NOTE 2 An Enumeration should be used only when it is clear that there will be no extensions, otherwise a Codelist should be used.

Integer

UInt8- One 8-bit byte unsigned computer representation of a whole number

UInt16 Two 8-bit bytes (16 bits) unsigned computer representation of a whole number

UInt32- Four 8-bit bytes (32 bits) unsigned computer representation of a whole number

UInt64- Eight 8-bit bytes (64 bits) unsigned computer representation of a whole number

Int8- One 8-bit byte (8 bits)signed computer representation of a whole number

Int16- Two 8-bit bytes (16 bits) signed computer representation of a whole number

Int32- Four 8-bit bytes (32 bits) signed computer representation of a whole number

Int64- Eight 8-bit bytes (64 bits) signed computer representation of a whole number

Real

This document uses the term, “Float” for single, not necessarily whole numbers. No fixed point representations are used. The “floats” may be either single (32 bit) or double (64 bit) precision. The alternative to “real” is a complex number, which is a 2-tuple, with the real part coming first and the imaginary part coming second.

URI
CharacterString containing a Uniform Resource Identifier corresponding to the W3C recommendation

URL
CharacterString containing a Uniform Resource Locator corresponding to the W3C recommendation

7. Conceptual Model Layout

The conceptual metadata model consists of a series of subsets, termed metadata sections, described using tables and used to organize related metadata elements. Following objectives adopted in the NMF, these sections (tables) are linked to other tables of elements to express the complex relationship between various sections of metadata. A given section of metadata may have a series of subsections related to it. This portion of the document describes the layout used throughout the remainder of this document to describe the methods used to illustrate the lidar metadata requirements.

7.1. Conceptual Model Structure Diagram

Conceptual model structure diagrams are used in the document to assist in the visualization between the various sections and subsections of metadata.

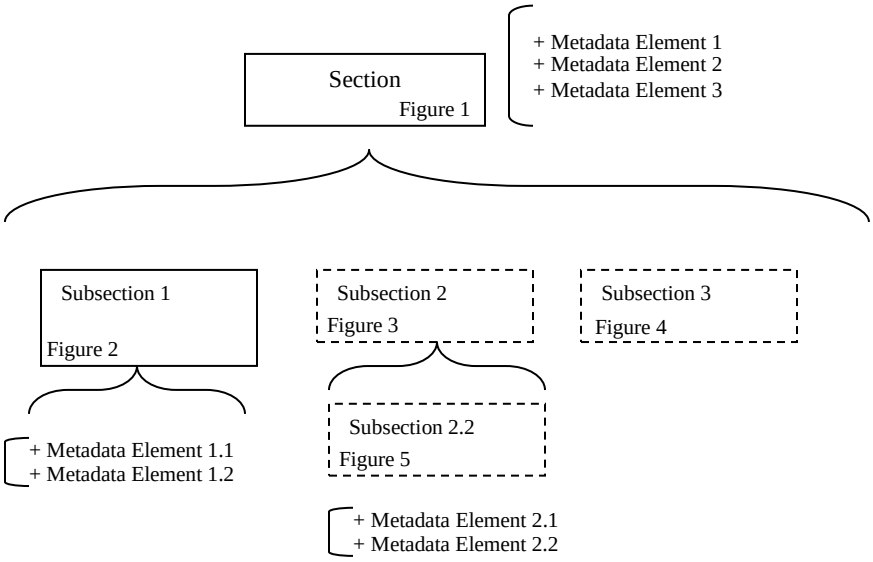


Figure 7- 1: --Conceptual Model Structure Diagram
The horizontal braces act as a "funnel"; those items below are elements of what is directly above.
Since the CMMD supports name-value reading and writing of data set content, every item has a unique name and multiplicity in a CMMD compliant data set by data tag is either 0 (See 9.8.), or exactly one for essential data items[10] and non-essential items chosen for population. See section 5.2. Solid rectangles represent the minimum element content to the level hierarchically above the over brace. The dotted rectangles indicate the possibility of horizontal or vertical extensions of the local data model hierarchy.

[10] Whereas the CMMD has no mandated data, profiles may mandate non-essential data items as well as the essential items. What is required is that there be uniqueness in the name-value mapping. A streaming solution with a limited memory client may find it convenient to repeat data-tags in the stream. This CMMD does not prevent this solution (though modern technology makes this an increasingly unneeded strategy). Note that a repeated name-value occurrence does not mean a multiplicity greater than one has been used. (It does not represent an overloading of the key, which is what is implied by the term "multiplicity".) The value must be the same if the name is the same. In implementations where the data set has been stored in a library or archive, repeating of data in the file is inefficient and a DAL option is to flag it as an error when it occurs.

In the conceptual model structure diagram (Figure 7- 1), the metadata sections are shown as rectangles. The rectangle includes a section name ("Section") and a figure number (figure 1) to allow the reader to identify additional figures associated with a diagram. Individual metadata elements associated with a given section are also indicated ("+ Metadata Element"). Figures of this form are found in section 9, Data Structure of the lidar Conceptual Model

7.2. Conceptual Model Metadata Section Tables

In conjunction with the Conceptual Metadata Structure Diagrams described above, Metadata Section tables will be provided for the metadata sections and subsections. The Metadata Section Tables will be organized as indicated in Table 1 below.

Table 1: Sample Metadata Section Table

Row	Tag Name/key	Brief Definition	Domain	Essential/not
1	Section 1	The first metadata section	Lines 2-5	
2	Item 1	The first metadata element		
3	Class 1	The first Class	Lines 4-5	
4	Item 2	The second metadata element		
5	Item 3	The third metadata element		

In the metadata section tables:

- Row indicates the row number within the section
- Tag Name/key identifies the name (hierarchically structured identifier) of the specific class or metadata element
- Brief Definition provides a short indicative definition for the class or metadata element
- Domain indicates the allowable value type for a given element. Domain is a selection of "number", "code" or "char string".
- Essential/Not carries the code "E" if the item is necessary for logical closure of the data in the data set. It carries the code "C" for essentially contingent, if the item is necessary when certain content is provided in the data set, but not necessary when that content is not present. At the conceptual level, nothing is required beyond the essential/contingent, as any exploitation support capability could be unfunded in some system to which this CMMD is applicable. For a self defining model, however, some items are essential to the data set navigability. Blanks in this field make the population discretionary to the organization sponsoring the creation of CMMD compliant data sets. The more that *applicable* fields are populated in a data set, the more capability the data set can support. The word "applicable" was emphasized to note that the dictionary in this document is overcomplete with respect to any single lidar data product.

Consequently, any lidar product in the VRP phenomenology class would be some profile, or subset, of the content of this CMMD.

In the example in table 1 above, row 1 indicates the overarching class that defines this metadata section. The domain indicates that rows 2-5 are indicated in this class. The second row provides one metadata element. The third row indicates a subclass within the metadata section that consists of rows four through five. Row four indicates a metadata element within class one. Row five indicates another metadata element that can occur within class one. Italics are used in this illustration because all rows are referenced in this discussion, but in the actual metadata tables in section 9, only populatable rows are numbered so that the number of defined data items is apparent.

8. lidar Conceptual Model Overview

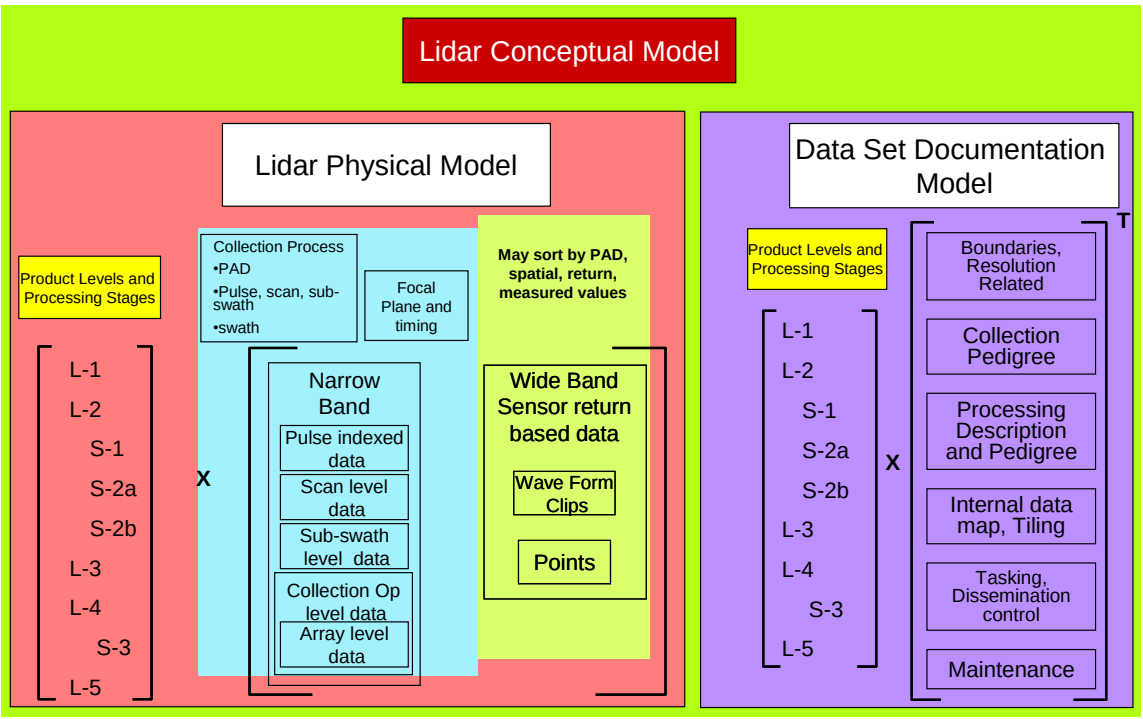


Figure 8- 1: The lidar conceptual model is a composite of a lidar Physical model and a data set documentation model

8.1. Overview of the Physical Model of Lidar VRP Class

8.1.1 Sub-models

8.1.1.1 Timing Model [“focal plane and timing” box in Figure 8-1]

While volume return products can be created from continuous wave laser scanning devices[11], in order to describe collection and processing of LIDAR sensors, a timing model is required. The principle LIDAR ranging technique amounts to measuring the roundtrip time of flight for transmitted pulses of many photons, even when only a handful of those photons may be in the received echo. Section 10.1.1 gives the detailed model of the documentation of the round trip time interval, which maps into a range measurement via a propagation model for light in the medium between sensor and target scene.

8.1.1.2 Beam Direction Control Model [scan part of collection process box in 8-1]

Normal Laser sources naturally have a narrow beam, both in the size of the resonating cavity exit aperture and in the divergence angle of that emerging beam. Consequently, in order to cover significant amounts of ground some method of spreading and/or scanning the illuminating laser beam is necessary. The position of a scatterer is determined from the fundamental measurement of a time and an angle for

[11] These are not “time to range” lidars, but rather laser driven parallax devices. They are most often used at short ranges.

received photons[12]. Where within a beam the energy of a return arrives and where the beam was pointed combine to give an angular coordinate to the "ray" from the sensor to scatterer. The round trip time, as noted above, can be converted to a range and so the position of a scatterer relative to the sensor can only be determined via beam descriptor data. A model for the sensitive directions within the beam and the beam direction as a function of time is given in section 10.1.2.

8.1.1.3 Sensor Measurements, Hierarchical organization (PCD), Scan, Subswath and Swath [see collection process box in figure 8-1]

A lidar sensor collects data in temporally associated individual measurements by **p**ulse. The number of individual photon flux measurements or arrival times collected for a single pulse depends on how many individual reporting sensing elements there are on the focal plane sensing component, which we designate as an array. There is only one array per **c**hannel for a given pulse. The sensor may expose and read out more than one array per received pulse. Each individual reporting sensing element has its own cone of optical sensitivity aligned in a look **d**irection (which we call a ray). So there is a natural hierarchy of data organization of pulse, array (via the choice of channel) and array element that leads to a single **d**irection's worth of time history measurement for every channel for each pulse. The most fundamental raw data division is the single pulse, single channel, and single direction/element (PCD) time history. Thus, a single measurement point position in the scene space leads to a data parameterization in terms of a single value for each of the following:

- Pulse index
- Channel index
- Direction index within the channel's array

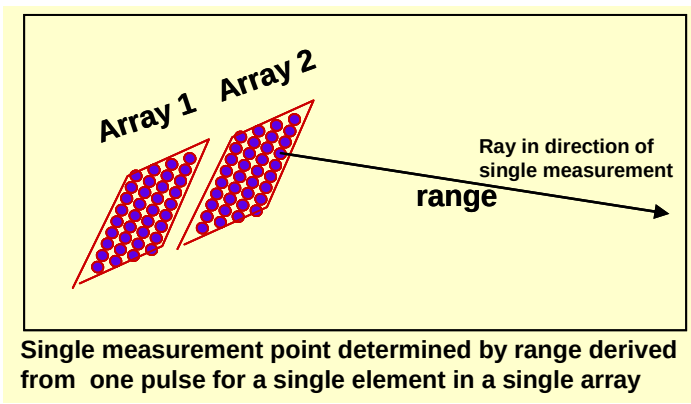


Figure 8.1- 1 Look direction associated with array and element choice

In a linear mode lidar, there may be a sequence of times at which a current proportional to received photon flux is measured. In a Geiger mode lidar, the time history is the latched[13] clock reading of the moment of the first detected photon arrival since the array was last quenched.

[12] This a statement that applies to normal, real aperture LIDAR. Synthetic aperture lidar is also a permitted mode and in that case a scene model substitutes for angle data in determining the position of a scene return. But even for SAL, the beam pattern and beam scanning data can be key inputs to the processing algorithm.

[13] "Latching" is the process of capturing a time reading as if the clock were stopped at the moment of an event. It is a figurative term. In such systems as lidars, the fast clocks are not stopped, but rather the value of a count corresponding to an

While the precise setting of the beam directing mechanism as a function of time is used to compute the positions of scatterers that are the sources of receive light 'echoes', the articulations of the beam direction are generally repeated periodic patterns. Multiple pulses can be aggregated over a single period of beam articulation that is called a "scan". The projection of the beam's scan pattern to the ground reflects the combination of beam articulations and sensor platform movement during a scan. For very high frequency pulsing of the illuminator (thousands of Hz), moderately high scan frequencies may be achieved (tens to hundreds of Hz). Groups of scans may be collected into subswaths. There are two ways this may occur.

The first way for collecting scans into a subswath is a natural consequence of the commanded lay down pattern of scans. This is the "sensor based" subswath grouping and an example might be the case where the scan pattern is a group of nominally straight scans parallel to the track of the platform. [See Figure 10.1.4-4] Only a few such consecutive scans can all have common along track coverage. The next scan after a group that does have overlapping along track coverage would have no common projected coverage with the first scan of the previous group, and so could logically be said to be the first scan of the next group of scans comprising a "subswath."

The second way subswath aggregation of scans can be done is processor based. An arbitrary number of scans might be collected into a single subswath so that parallel processing can divide the totality of collected scans into a group whose size is limited by memory management considerations. Another criterion that may be adopted in deciding how many consecutive scans to treat as a subswath is the need for adaptive registration. If it is known, a'priori, that a single adaptive registration can lead to required geometric fidelity of a product that has been blocked in groups of some number of scans, that level of collection of scans could be designated as a subswath.

Subswaths once formed, can be adaptively registered (application of block adjustments) before the final aggregation into a swath, which is the level of data aggregation associated with a single lidar collection operation. A swath is normally associated with the flight line of the sensor platform, and is associated with a single setting of the sensor's collection parameters (such as scan control, PRF, Focal plane sensitivity, range timing resolution,…). That is, a swath is the dataset product level coverage associated with a single collection operation. The data relating these levels of aggregation for lidar is described in sections 10.1.3.

The data at pulse, scan, subswath, and swath levels is *logically* connected within those data divisions. For the purpose of *physically* blocking the data for I/O management, it is convenient to collect the data into "tiles". Section 10.2.1 also explains how this conceptual model supports data tiling within large data sets.

---

event is put onto a data bus to be recorded. The latching is the stopping of the *counter* at the increment when the photon triggers the diode output.

8.2. Overview of the Lidar Conceptual Model Data Relations.

8.2.1 Top Level (Class) Data Categories

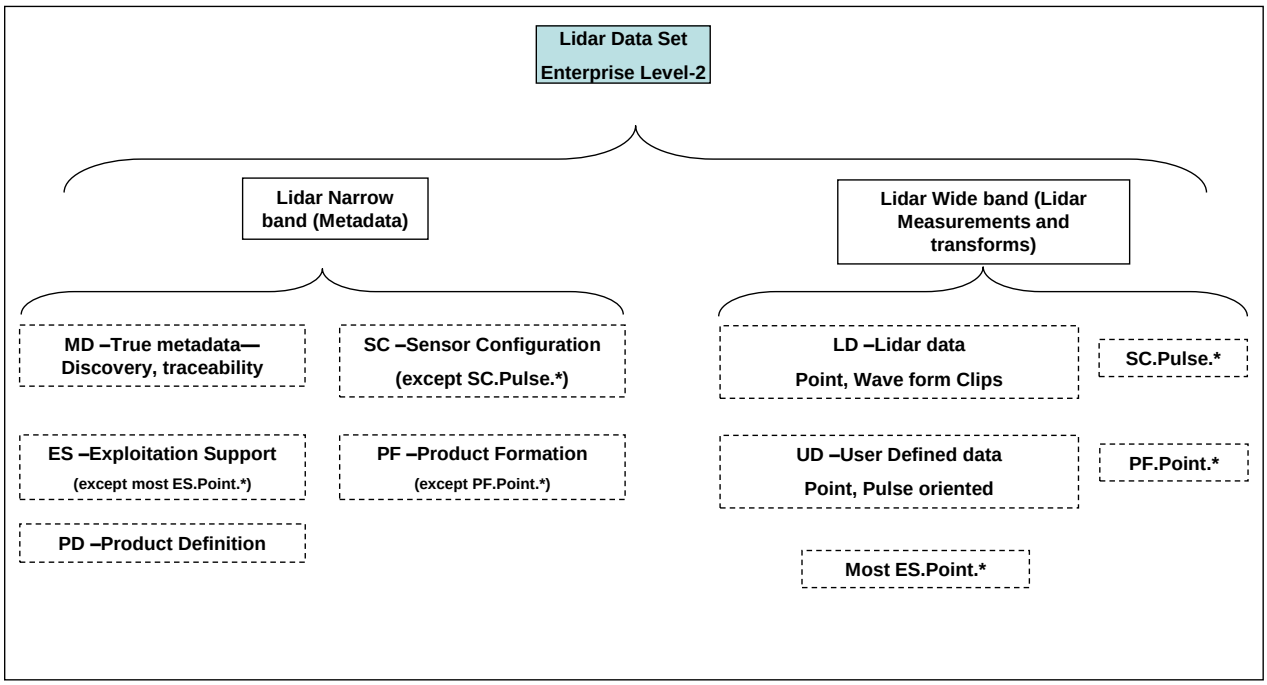


Figure 8.2- 1: Overview of the lidar Conceptual Model Data Documentation Structure

As indicated above, the lidar conceptual model divides the data into the following seven primary sections by "class": MD.*, ES.*, PF.*, PD.*,SC.*, LD.* and UD.*. At the property level, the data can also be divided into narrowband and wideband data groups. Most *.Point.* entities are wideband.

The MD.* incorporates the following NMF topics:

- Quality Metric Information – Provides the ability to store quality information on the lidar data set. This information can include positional accuracy, but is not limited to accuracy. It could contain other metric information such as spatial resolution, range resolution, etc.
- Source Acquisition – Provides insight into the sensor(s) and the platform(s) used in the data collection.
- Maintenance Information – Provides information on the party that is responsible for maintaining and updating the data set.
- Resource Identification – Provides important information related to the data set to include a description of the data set, the name of the party responsible for the data set, file creation information, information on aggregate data sets, security constraints, and legal constraints.
- Resource Lineage – Provides information on the resource (the data set). Will track items such as the processing that has been performed on the data set and who has performed the processing.
- Metadata Identification – Provides information related to the metadata including the type of metadata, who is responsible for the metadata, and security related to the metadata.

The ES.* incorporates the following NMF related topics:

- Reference System – Provides information on both the horizontal and vertical reference frames used in the generation of the point record information. While the CMMD employs only local non-geodetic (Cartesian[14]) frames for point clouds, specifying these involves offsets, orientations and scaling factors
- uncertainty information
- Wideband ES.Point.* gives information at the point level such as the return and channel

The PF.* incorporates the following topics:

- Noise reduction, internal registration, time to position transformation, file regeneration from precursor data, sorting information, narrowband user defined data
- Wideband PF.Point.* gives information on synthetic point origins

The PD.* incorporates the following topics:

- Mapping of the product's tiles, array sizing, word sizing, data volume reduction,

The SC.* incorporates the following topics:

- Items such as sensor settings, sensor trajectory/orientation and detailed records of data collection

The LD.* incorporates the following topics:

- Wideband lidar data

The UD.* incorporates the following topics:

- Wideband user defined data

[14] The frames are Cartesian (right handed sets of 3 orthogonal axes), but the coordinates may be Cartesian or angle and range based

9. Data Structure of the Lidar Conceptual Model

Section 7 provided a general overview of the lidar conceptual model and the associated sections. This section provides a more detailed description of the model's data structure using a combination of Model Structure diagrams and associated metadata tables. For detailed definitions of the metadata elements, see the dictionary in Appendix 20.

9.1. MD.* True Metadata

The term, "true metadata" is used here because this is actually data about the data. The metadata elements defined by the MD.* are for discovery/selection, dissemination control and usage constraint purposes. Nothing in this section should be used in manipulating or processing the data in either tools or downstream product generation applications.

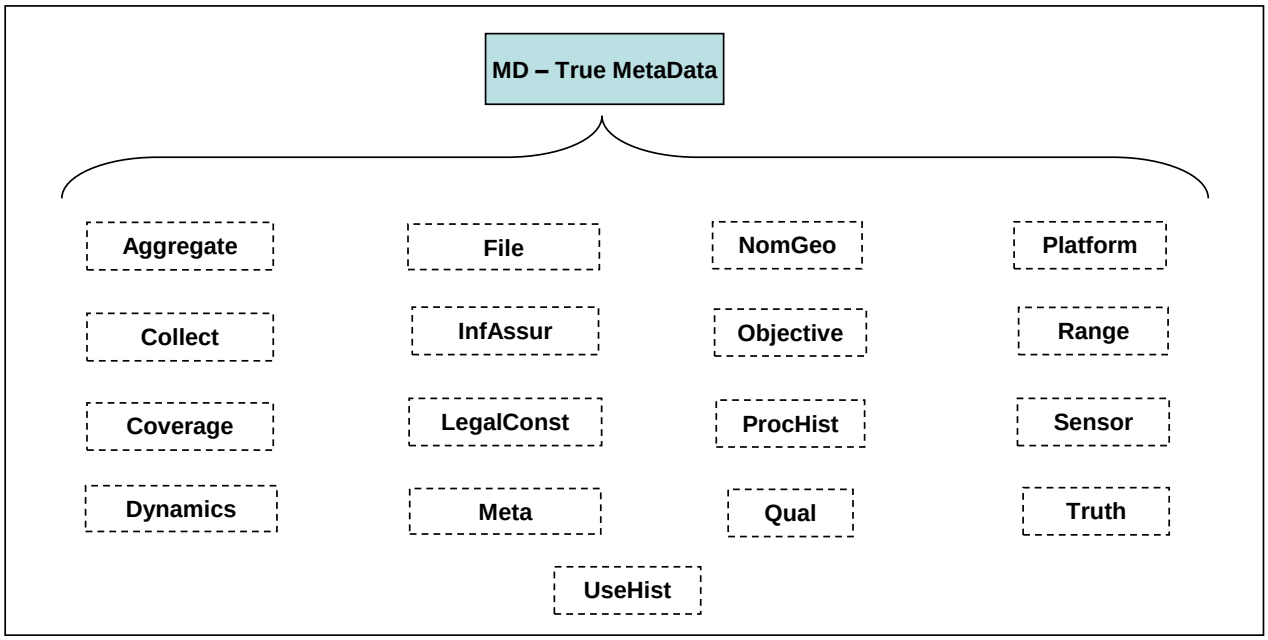


Figure 9.1- 1: MD -True Metadata – Second Level (Class) Structure

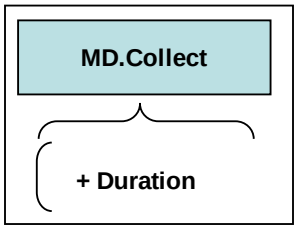


Figure 9.1- 2: MD.Collect Metadata Elements

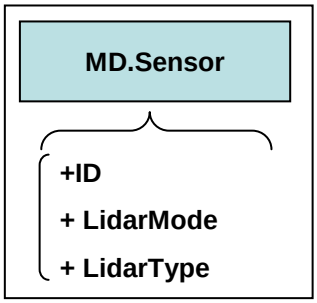


Figure 9.1- 3: MD.Sensor Metadata Elements

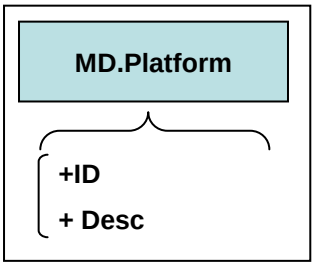


Figure 9.1- 4: MD.Platform Metadata Elements

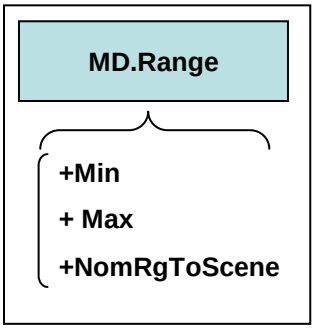


Figure 9.1- 5: MD.Range Metadata Elements

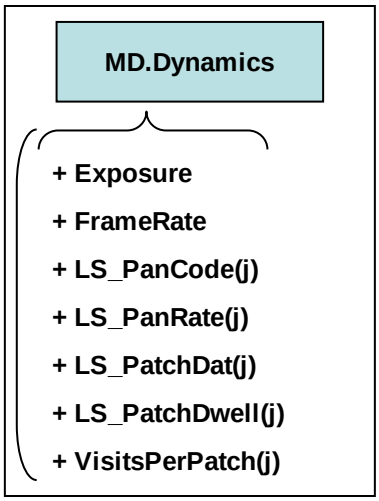


Figure 9.1- 6: MD.Dynamics Metadata Elements

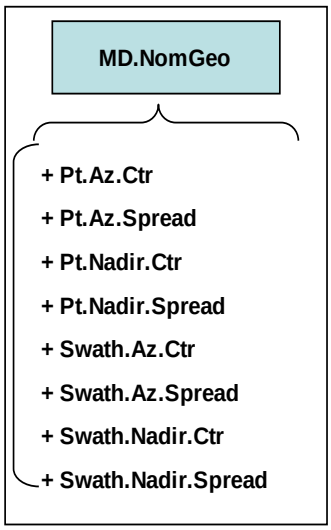


Figure 9.1- 7: MD.NomGeo Metadata Elements

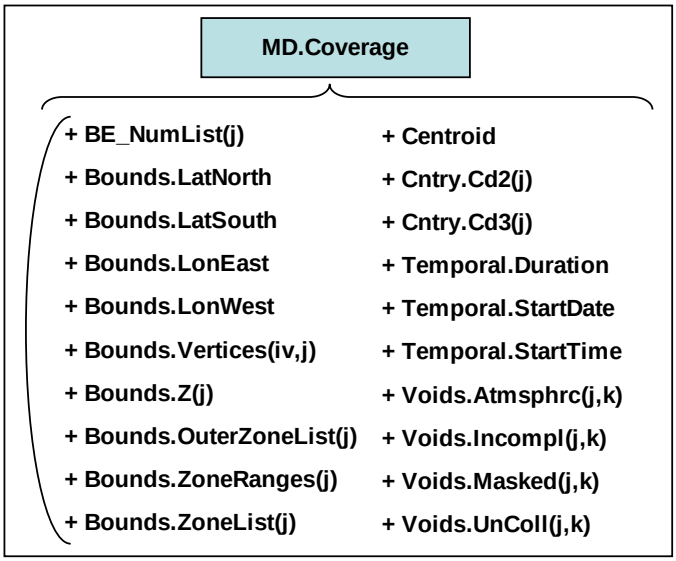


Figure 9.1- 8: MD.Coverage Metadata Elements

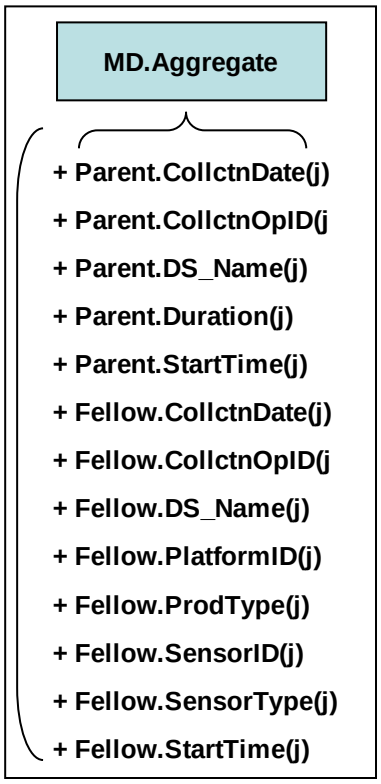


Figure 9.1- 9: MD.Aggregate Metadata Elements

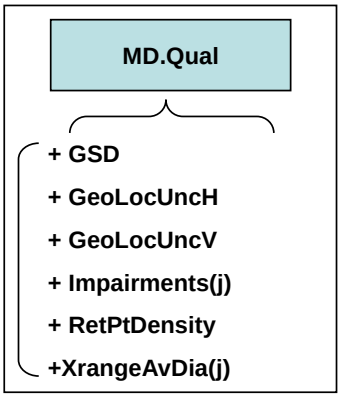


Figure 9.1- 11: Qual Elements for Metadata

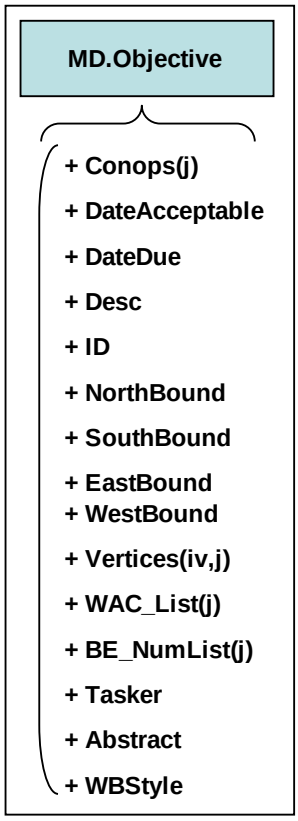


Figure 9.1- 10: Objective Elements for Metadata

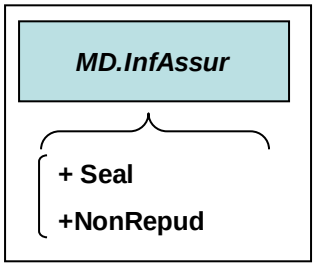


Figure 9.1- 12: Candidate Information Assurance Elements for Metadata

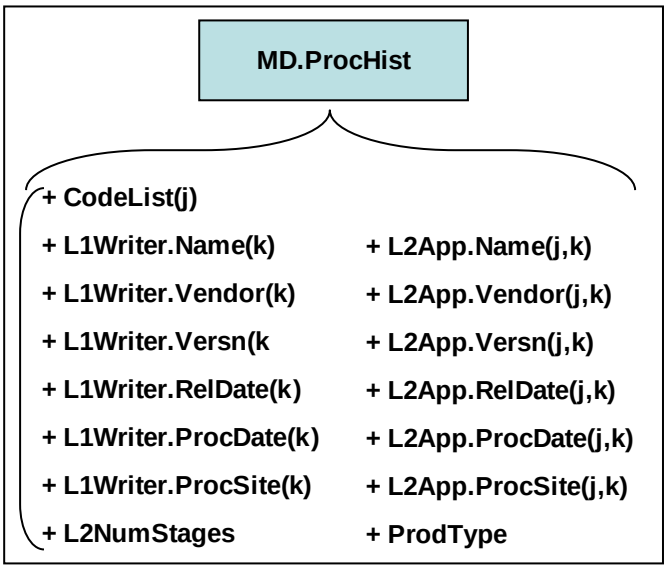


Figure 9.1- 13: ProcHist Elements for Metadata

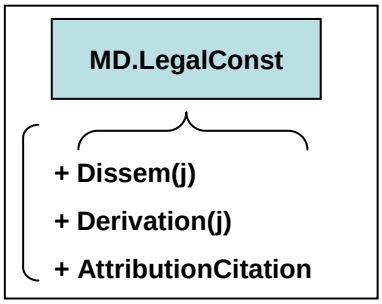


Figure 9.1- 14: LegalConst Elements for Metadata

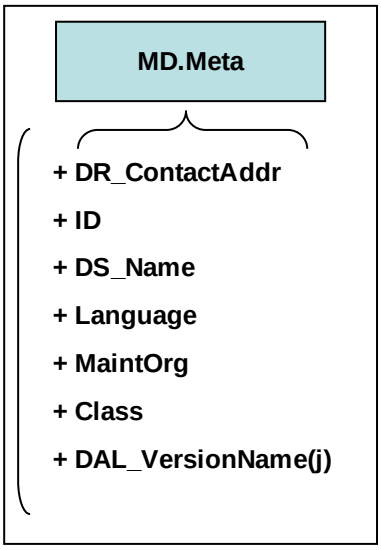


Figure 9.1- 15: Meta Elements for Metadata

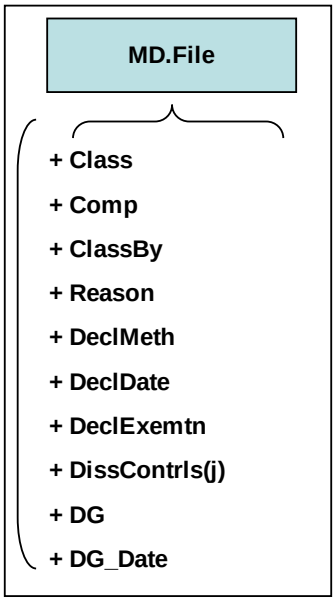


Figure 9.1- 16: File Elements for Metadata

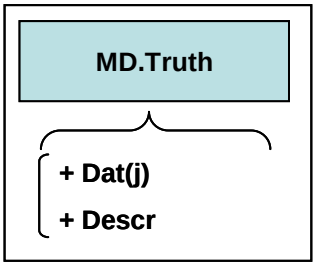


Figure 9.1- 17: Truth Elements for Metadata

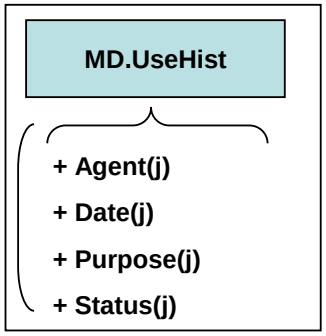


Figure 9.1- 18: UseHist Elements for Metadata

In the Metadata tables, the "Domain" heading is for coded, text (char string) or engineering value (number). The "Essential/not" heading indicates whether the data set can logically function without the data. Generally only PD.* items are essential. The data definition does not set population requirements. That is left to sponsoring organizations. Data which is available in principle may not be affordable. So, however advisable population of a data item on a routine basis is, obligation to populate at this level of documentation is based only on logical closure.

Table 2: MD.* --True Metadata

Table Line #	Tag Name	Brief Definition	Domain	Essential/not
	MD.*	True Metadata class of data	1-121	
	MD.Aggregate.*	"Aggregate" property of MD Items associated with related data sets	1-13	
1.	MD.Aggregate.Parent.CollctnDate(j)	is the Truncated Julian Day (TJD) of collection of data in the parent data sets. For L-3 and lower data sets, this date is the same as that of the current data set.	**number**	
2.	MD.Aggregate.Parent.CollctnOpID(j)	provides the identifier(s) for the collection operation(s) for the parent data set(s) of the current data set.	**Char string**	
3.	MD.Aggregate.Parent.DS_Name(j)	identifies the data set(s) from which the current data set is derived.	**Char string**	
4.	MD.Aggregate.Parent.Duration(j)	is the difference between the first collected data element's collection time and the last collected data element's collection time for the jth parent data set.	**number**	
5.	MD.Aggregate.Parent.StartTime(j)	indicates the times of collection start for the parent data sets of the current data set.	**number**	
6.	MD.Aggregate.Fellow.CollctnDate(j)	is the date of collection of data in the Fellow data products, expressed as Truncated Julian Day.	**number**	
7.	MD.Aggregate.Fellow.CollctnOpID(j)	provides the identifiers(s) for the collection operations for the fellow data set(s) of the current data set.	**Char string**	
8.	MD.Aggregate.Fellow.DS_Name(j)	identifies Fellow data sets of the current data	**Char**	

		set.	**string**	
9.	MD.Aggregate.Fellow.PlatformID(j)	identifies the platform(s) from which data for the cited fellow data sets have collected.	**Char string**	
10.	MD.Aggregate.Fellow.ProdType(j)	indicates the product level and stage of Fellow data sets.	**code**	
11.	MD.Aggregate.Fellow.SensorID(j)	identifies the sensor ID(s) that collected data for the cited fellow data sets.	**Char string**	
12.	MD.Aggregate.Fellow.SensorType(j)	indicates the type of sensor, e.g. LIDAR, EO color, Video MWIR, … used for Fellow data set collection.	**code**	
13.	MD.Aggregate.Fellow.StartTime(j)	indicates the times of collection start for the cited fellow data sets of the current data set.	**number**	
	MD.Collect.*	"Collect" property of MD Items associated with the full collection operation	14-14	
14.	MD.Collect.Duration	gives the length of the temporal interval spanned by the collection operation from which the current data set has been formed.	**number**	
	MD.Coverage.*	Coverage property of MD Items associated with the geospatial extent captured in the data set.	15-34	
15.	MD.Coverage.BE_NumList(j)	is the list of Basic Encyclopedia numbers covered by the current data set.	**Char string**	
16.	MD.Coverage.Bounds.LatNorth	contains the northern most latitude of the coverage area	**number**	
17.	MD.Coverage.Bounds.LatSouth	contains the southern most latitude of the coverage area	**number**	
18.	MD.Coverage.Bounds.LonEast	contains the eastern most longitude of the coverage area	**number**	
19.	MD.Coverage.Bounds.LonWest	contains the western most longitude of the coverage area	**number**	
20.	MD.Coverage.Bounds.ZoneList(j)	is a vector indentifying all of the MGRS GZDs intersecting the current data set.	**code**	
21.	MD.Coverage.Bounds.OuterZoneList(j)	lists the four spatial-extent-defining MGRS GZDs.	**code**	
22.	MD.Coverage.Bounds.ZoneRanges(j)	contains the number of MGRS GZDs spanning the latitude and longitude extents of the current data set.	**number**	
23.	MD.Coverage.Bounds.Vertices(iv,j)	counter clockwise list of vertices beginning at North East most for the full data set iv=vertex index j=1-->Lat j=2-->Lon	**number**	
24.	MD.Coverage.Bounds.Z(j)	j=1 min Z, j=2 max Z	**number**	
25.	MD.Coverage.Centroid	centroid of data set lat, long, Z	**number**	
26.	MD.Coverage.Cntry.Cd2(j)	List of 2 letter country codes Need to add index range j=1 to num countries in data set	**code**	
27.	MD.Coverage.Cntry.Cd3(j)	List of 2 letter country codes Need to add index range j=1 to num countries in data set	**code**	
28.	MD.Coverage.Temporal.Duration	gives the length of the temporal coverage interval of the present data set.	**number**	
29.	MD.Coverage.Temporal.StartDate	gives the day of collection for the earliest items in the current data set.	**number**	
30.	MD.Coverage.Temporal.StartTime	gives the seconds after midnight Time of the earliest data in the current data set.	**number**	

31.	MD.Coverage.Voids.Atmsphrc(j,k)	gives the vertices for a list of geographic regions within the outer boundary of the data set known to have absent or degraded data due to atmospheric losses.	**code**	
32.	MD.Coverage.Voids.Incompl(j,k)	gives the vertices for a list of geographic regions within the outer boundary of the data set known to have only partial data collection due to unavoidable and predictable collection mode behavior.	**code**	
33.	MD.Coverage.Voids.Masked(j,k)	gives the vertices for a list of geographic regions within the outer boundary of the data set known to be in the lidar shadow of terrain features.	**code**	
34.	MD.Coverage.Voids.UnColl(j,k)	gives the vertices for a list of geographic regions within the outer boundary of the data set known to be uncollected bands within the data set.	**code**	
	MD.Dynamics	"Dynamics" property of MD items associated with determining the temporal change scales exploitable in the dataset	35-41	
35.	MD.Dynamics.Exposure	gives the time to collect one look of a **L**ocal **S**cene patch in the dataset.	**number**	
36.	MD.Dynamics.FrameRate	determines the number of times per second that a **L**ocal **S**cene patch is presented in the dataset.	**number**	
37.	MD.Dynamics.LS_PanCode(j)	describes the type of panning present in the dataset.	**code**	
38.	MD.Dynamics.LS_PanRate(j)	gives indicative values of the pan rate for the dataset.	**number**	
39.	MD.Dynamics.LS_PatchDat(j)	describes **L**ocal **S**cene patches in the dataset.	**number**	
40.	MD.Dynamics.LS_PatchDwell	MD.Dynamics.LS_PatchDwell characterizes how many seconds **L**ocal **S**cene patches persist in the dataset.	**number**	
41.	MD.Dynamics.VisitsPerPatch(j)	indicates local region revisits that may occur within the dataset.	**number**	
	MD.File.*	"File" property of MD Items associated with security handling of the data set file.	42-51	
42.	MD.File.Class	Classification of NB + WB together	**Char string**	
43.	MD.File.ClassBy	identifies the original classification authority for the content of the data set.	**code**	
44.	MD.File.Comp	caveats	**code**	
45.	MD.File.DeclDate	Declass date	**code**	
46.	MD.File.DeclExemtn	identifies applicable declassification exemption.	**code**	
47.	MD.File.DeclMeth	contains declassification method instructions.	**code**	
48.	MD.File.DissContrls(j)	MD.File.DissContrls(j) contains dissemination control codes.	**code**	
49.	MD.File.DG	Downgrade	**code**	
50.	MD.File.DG_Date	Downgrade Date	**number**	
51.	MD.File.Reason	contains the reason(s) for classification.	**code**	

	MD.InfAssur	"InfAssur property of MD Items to assure information integrity of data sets	52-53	
52.	MD.InfAssur.Seal	is an information assurance parameter meant to act as a security seal that can be used to detect alterations to a CMMD compliant file after authoritative generation.	**code**	
53.	MD.InfAssur.NonRepud	is an information assurance parameter meant to act as a non-repudiation guarantee that can be used to authenticate the originator's assurance of file origin.	**code**	
	MD.LegalConstr.*	"LegalConstr" property of MD Items associated with the legal constraints for use of this data set.	54-56	
54.	MD.LegalConstr.AttributionCitation	How to cite for attribution	**code**	
55.	MD.LegalConstr.Derivation(j)	codes for distribution of derived products	**code**	
56.	MD.LegalConstr.Dissem(j)	Legal Constraints codes for legal content protection governing secondary distribution copyright, license, trademark, patent,…	**code**	
	MD.Meta.*	"Meta" property of MD Items associated with documentation of metadata for this data set	57-63	
57.	MD.Meta.Class	Highest Classification of any meta data item when WB content is not present.	**Char string**	
58.	MD.Meta.DAL_VersionName(j)	string indicating which set of DAL modules can read this file. j=1 to number of DAL routines known to successfully read current file.	**Char string**	
59.	MD.Meta.DR_ContactAddr	URL where open DRs can be viewed, New DRs can be submitted	**Char string**	
60.	MD.Meta.DS_Name	Name of this data set up to system appended qualifiers	**Char string**	
61.	MD.Meta.ID	Indicates the data set is governed by the lidar Data Definition and Standard doc. E.g. "Volumetric class lidar Conceptual model"	**Char string**	E
62.	MD.Meta.Language	Code:use International Latin alphabet language code list Data set character strings are in this language.	**code**	
63.	MD.Meta.MaintOrg	Org maintaining the metadata definition doc. +contact phone or URL	**Char string**	
	MD.NomGeo.*	"NomGeo" property of MD Items associated with the collection geometry for the data set	64-71	
64.	MD.NomGeo.Pt.Az.Ctr	is the central value of the collected range of azimuth angles on a point basis and is intended to characterize point level collection geometry aspect.	**number**	
65.	MD.NomGeo.Pt.Az.Spread	is the range of azimuth angle about the central value of the collected azimuth angles for each point.	**number**	
66.	MD.NomGeo.Pt.Nadir.Ctr	is the central value of the collected range of nadir angles on a point basis and is intended to characterize point level collection geometry aspect.	**number**	
67.	MD.NomGeo.Pt.Nadir.Spread	is the range of Nadir angle about the central value of the collected Nadir angles for each point.	**number**	

68.	MD.NomGeo.Swath.Az.Ctr	is the central value of the collected range of azimuth angles for the swath	**number**	
69.	MD.NomGeo.Swath.Az.Spread	is the range of azimuth about the central value of the collected azimuth angles over the collected swath.	**number**	
70.	MD.NomGeo.Swath.Nadir.Ctr	is the central value of the collected range of Nadir angles for the full swath contained in the data set.	**number**	
71.	MD.NomGeo.Swath.Nadir.Spread	is the range of Nadir angle about the central value of the collected Nadir angles across the collected swath.	**number**	
	MD.Objective.*	"Objective" property of MD Items associated with the objectives of the data collection	72-86	
72.	MD.Objective.Abstract	summary of what the data set is	**Char string**	
73.	MD.Objective.BE_NumList(j)	Basic Encyclopedia number list of collection requirement	**Char string**	
74.	MD.Objective.Conops(j)	provides a list of codes that describe the conops for collecting and processing the data in the current data set.	**code**	
75.	MD.Objective.DateDue	Date collection must be done before	**number**	
76.	MD.Objective.DateAcceptable	date that collection may be subsequent to	**number**	
77.	MD.Objective.Desc	eg "campaign to map region ABC"	**Char string**	
78.	MD.Objective.EastBound	campaign objective boundary on East	**number**	
79.	MD.Objective.ID	e.g. a collection requirement number	**code**	
80.	MD.Objective.NorthBound	campaign objective boundary on North	**number**	
81.	MD.Objective.SouthBound	campaign objective boundary on South	**number**	
82.	MD.Objective.Tasker	Entity submitting requirement	**Char string**	
83.	MD.Objective.Vertices(iv,j)	counter clockwise list of vertices beginning at North East most of fully tasked area	**number**	
84.	MD.Objective.WAC_List(j)	World Aeronautical Chart Cell list of regions in objective campaign	**code**	
85.	MD.Objective.WBStyle	code for pts, waveforms, waveforms + pts, TIN, BE DEM,…	**code**	
86.	MD.Objective.WestBound	campaign objective boundary on West	**number**	
	MD.Platform.*	"Platform" property of MD Items associated with the collection platform	87-88	
87.	MD.Platform.Desc	Description of platform, eg "International Space station", or "prop plane", etc	**Char string**	
88.	MD.Platform.ID	Eg tail number, Sat ID (e.g. IceSat, Calipso), pedestal number for ground fixed platform	**Char string**	
	MD.ProcHist.*	"ProcHist" property of MD Items associated with the processing history leading to the current data set.	89-103	
89.	MD.ProcHist.CodeList(j)	is a list of product type descriptions that are referenced to describe the current product and its precursors.	**code**	
90.	MD.ProcHist.L1Writer.Name	L1 Writer Application Name	**Char string**	
91.	MD.ProcHist.L1Writer.ProcDate	L1 written on this date TJD, displayable as YYYYMMDD	**number**	
92.	MD.ProcHist.L1Writer.ProcSite	L1 written at this processing site	**Char**	

			string	
93.	MD.ProcHist.L1Writer.RelDate	L1 S/W release date TJD displayable as YYYYMMDD	**number**	
94.	MD.ProcHist.L1Writer.Vendor	Name of vendor providing the application for writing L1	**Char string**	
95.	MD.ProcHist.L1Writer.Versn	L1 S/W Version Num XX.XX	**Char string**	
96.	MD.ProcHist.L2App.Name(j)	j=1 to L2NumStages Application name for stage j of processing	**Char string**	
97.	MD.ProcHist.L2App.ProcDate(j)	j=1 to L2NumStages Application execution date for stage j of processing	**number**	
98.	MD.ProcHist.L2App.ProcSite(j)	j=1 to L2NumStages Application processing sitefor stage j of processing	**Char string**	
99.	MD.ProcHist.L2App.RelDate(j)	j=1 to L2NumStages Application release date for stage j of processing	**number**	
100.	MD.ProcHist.L2App.Vendor(j)	j=1 to L2NumStages Application vendor for stage j of processing	**Char string**	
101.	MD.ProcHist.L2App.Versn(j)	j=1 to L2NumStages Application version for stage j of processing	**Char string**	
102.	MD.ProcHist.L2NumStages	Number of L-2 stages with separate apps executed in making the current data set	**number**	
103.	MD.ProcHist.ProdType(j)	indicates the enterprise product level and processing stage of the stored products in the thread ending in the current dataset. This is a discovery level summary of the processing objectives performed to create the current data set.	**code**	
	MD.Qual.*	"Qual" property of MD Items associated with the data set quality	104-109	
104.	MD.Qual.GeoLocUncH	Nominal meters of error volume linear dim horizontal	**number**	
105.	MD.Qual.GeoLocUncV	Nominal meters of error volume linear dim Vertical	**number**	
106.	MD.Qual.GSD	Gives a representative value over the product of the geometric mean of sensor pixel major and minor axis lengths as projected to the ground.	**number**	
107.	MD.Qual.Impairments(j)	reports codes for a set of degradation categories.	**code**	
108.	MD.Qual.RetPtDensity	mean number of returns reported per cubic meter over data set.	**number**	
109.	MD.Qual.XrangeAvDia(j)	gives the least and greatest values of the effective cross range averaging circle diameter over the product.	**number**	
	MD.Range.*	"Range" property of MD Items associated with the range at which data was collected	110-112	
110.	MD.Range.Max	see min range	**number**	
111.	MD.Range.Min	This may be sensitive data, so it can't be mandatory. It is of virtually no value. Signal to noise related items can be provided independently of the range and geolocation errors that are range dependent can have the range already incorporated. Would be	**number**	
112.	MD.Range.NomRgToScene	Nominal range to scene. This could be the altitude of a plane, distance from pedestal to citiscape,…	**number**	

	MD.Sensor.*	"Sensor" property of MD Items associated with the sensor	113-115	
113.	MD.Sensor.ID	e.g. a mission number. Uniquely identifies the series and series number of the sensor	**Char string**	
114.	MD.Sensor.lidar Mode	Code for Demodulation scheme (linear mode detector, Geiger mode detector, heterodyne mixer, polarization mixer,…	**code**	
115.	MD.Sensor.lidarType	Code for FP design (APD, MCP,StreakTube,PMT, RF modulated CW…	**code**	
	MD.Truth.*	Truth property of MD Descriptions of ground truth data available for fine calibration	116-117	
116.	MD.Truth.Dat(j)	carries a list of codes defining the ground truth data incorporated in the lidar Data set	**code**	
117.	MD.Truth.Descr	provides a text description of the ground truthing mechanism when ground truthed returns are in the data set.	**Char string**	
	MD.UseHist.*	items associated with characterizing the use by human or software agents of the data in the current dataset	118-121	
118.	MD.UseHist.Agent(j)	provides a list of human or software agents that accessed the current file for use.	**Char string**	
119.	MD.UseHist.Date(j)	provides a list of dates that the current file was accessed for use by human or software agents.	**number**	
120.	MD.UseHist.Purpose(j)	provides a list of objectives for the human or software agents that accessed the current file for use.	**Char string**	
121.	MD.UseHist.Status(j)	provides a use status condition for each access event for the current file.	**code**	

9.2. ES –Exploitation Support

Exploitation support metadata provides information needed in product manipulation, interpretation, mensuration and display.

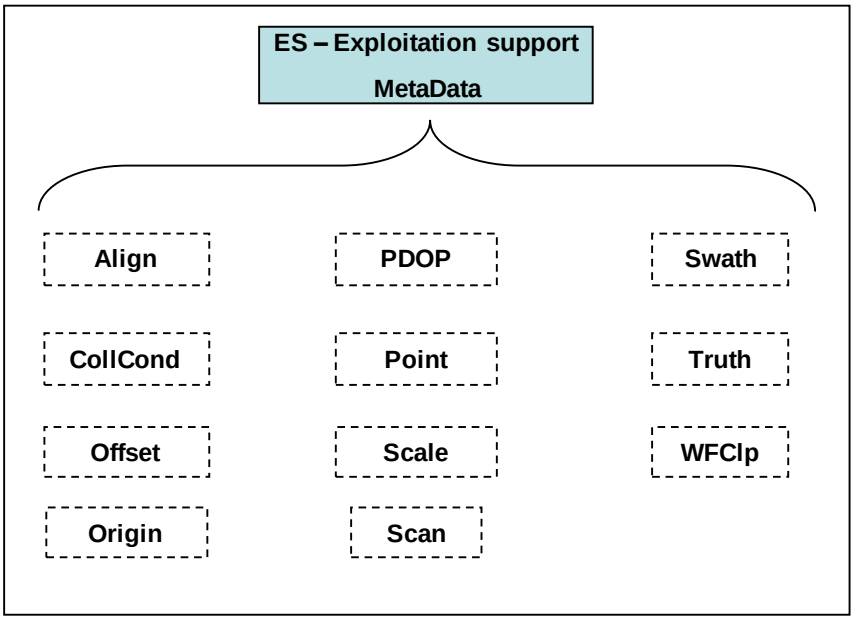


Figure 9.2- 1 ES – Exploitation support second level data structure

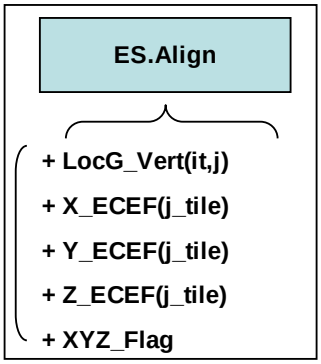


Figure 9.2- 2: Align elements for Exploitation support data

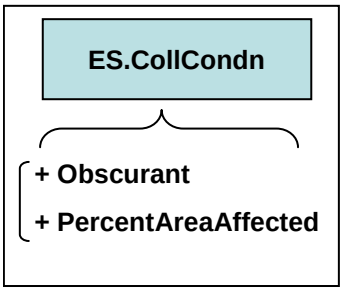


Figure 9.2- 3: CollCondn elements for Exploitation support data

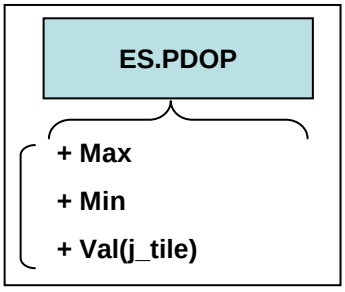


Figure 9.2- 4: PDOP elements for Exploitation support data

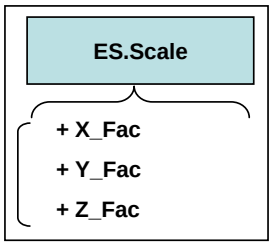


Figure 9.2- 5: Scale elements for Exploitation support data

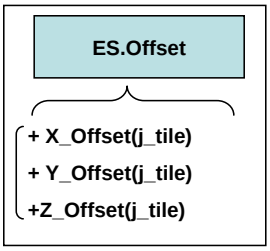


Figure 9.2- 6: Offset elements for Exploitation support data

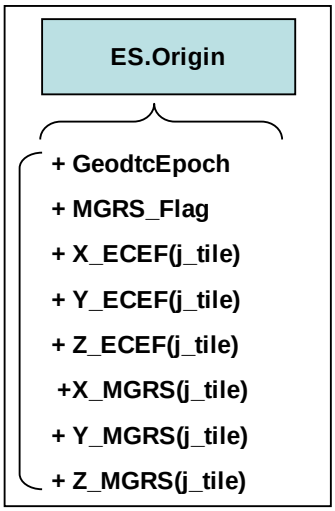


Figure 9.2- 7: Origin elements for Exploitation support data

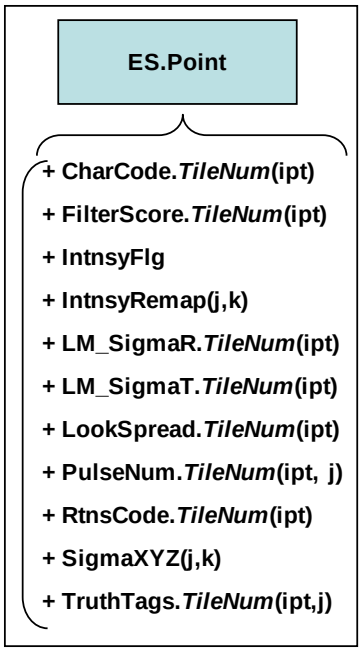


Figure 9.2- 8: Point elements for Exploitation support data

ES.Scan

+CorrParms.Alpha(j,k)	+CorrParms.XAlpha(j,k)	+CorrParmsT.Gamma(j,k)	+CorrParmsT.XGamma(j,k)	+RhoAlphaBeta(j)	+RhoZGamma(j)
+CorrParms.Beta(j)	+CorrParms.XBeta(j,k)	+CorrParmsT.X(j,k)	+CorrParmsT.YZ(j,k)	+RhoBetaGamma(j)	+SigmaAlpha(k)
+CorrParms.Gamma(j,k)	+CorrParms.XGamma(j,k)	+CorrParmsT.Y(j,k)	+CorrParmsT.YAlpha(j,k)	+RhoXAlpha(j)	+SigmaBeta(j)
+CorrParms.X(j,k)	+CorrParms.YZ(j,k)	+CorrParmsT.Z(j,k)	+CorrParmsT.YBeta(j,k)	+RhoXGamma(j)	+SigmaGamma(j)
+CorrParms.Y(j,k)	+CorrParms.YAlpha(j,k)	+CorrParmsT.R(j,n,k)	+CorrParmsT.YGamma(j,k)	+RhoXY(j)	+SigmaX(j)
+CorrParms.Z(j,k)	+CorrParms.YBeta(j,k)	+CorrParmsT.AlphaBeta(j,k)	+CorrParmsT.ZAlpha(j,k)	+RhoXZ(j)	+SigmaY(j)
+CorrParms.R(j,n,k)	+CorrParms.YGamma(j,k)	+CorrParmsT.AlphaGamma(j,k)	+CorrParmsT.ZBeta(j,n,k)	+RhoYAlpha(j)	+SigmaZ(j)
+CorrParms.AlphaBeta(j,k)	+CorrParms.ZAlpha(j,k)	+CorrParmsT.BetaGamma(j,k)	+CorrParmsT.ZGamma(j)	+RhoYBeta(j)	+SigmaNbar(j)
+CorrParms.AlphaGamma(j,k)	+CorrParms.ZBeta(j,n,k)	+CorrParmsT.XY(j,k)	+DelT_Cov	+RhoYGamma(j)	
+CorrParms.BetaGamma(j,k)	+CorrParms.ZGamma(j)	+CorrParmsT.XZ(j,k)	+GM_SigmaR(j)	+RhoYZ(j)	
+CorrParms.XY(j,k)	+CorrParmsT.Alpha(j,k)	+CorrParmsT.XAlpha(j,k)	+RhoAlphaGamma(j)	+RhoZAlpha(j)	
+CorrParms.XZ(j,k)	+CorrParmsT.Beta(j)	+CorrParmsT.XBeta(j,k)	+RhoXBeta(j)	+RhoZBeta(j)	

Figure 9.2- 9: Scan elements for Exploitation support data

ES.Swath

+CorrParms.Alpha(j,k)	+CorrParms.AlphaGamma(j,k)	+CorrParms.YAlpha(j,k)	+RhoAlphaGamma(k)	+RhoYAlpha(k)	+SigmaBeta(k)
+CorrParms.Beta(j,k)	+CorrParms.BetaGamma(j,k)	+CorrParms.YBeta(j,k)	+Rho XBeta(k)	+RhoYBeta(k)	+SigmaGamma(k)
+CorrParms.Gamma(j,k)	+CorrParms.XY(j,k)	+CorrParms.YGamma(j,k)	+RhoAlphaBeta(k)	+RhoYGamma(k)	+SigmaX(k)
+CorrParms.X(j,k)	+CorrParms.XZ(j,k)	+CorrParms.ZAlpha(j,k)	+RhoBetaGamma(k)	+RhoYZ(k)	+SigmaY(k)
+CorrParms.Y(j,k)	+CorrParms.XAlpha(j,k)	+CorrParms.ZBeta(j,n,k)	+RhoXAlpha(k)	+RhoZAlpha(k)	+SigmaZ(k)
+CorrParms.Z(j,k)	+CorrParms.XBeta(j,k)	+CorrParms.ZGamma(j)	+RhoXGamma(k)	+RhoZBeta(k)	+SensorBasedCoordFlg(j)
+CorrParms.R(j,n,k)	+CorrParms.XGamma(j,k)	+DelT_Cov	+RhoXY(k)	+RhoZGamma(k)	+ HiConfdncP_Fac(i,j,k)
+CorrParms.AlphaBeta(j,k)	+CorrParms.YZ(j,k)	+GM_SigmaRfrac(i,j)	+RhoXZ(k)	+SigmaAlpha(k)	+ PSF(j)
		+GM_SigmaT			

Figure 9.2- 10: Swath elements for Exploitation support data

ES.Truth

- + CalChRatios(ich)
- + IntnsyRef(itp,ich,j)
- + IllusKMZ.*j*
- + Pos(itp,j)
- + CovCorrs(itp, j)
- + CovSigs(itp,j)

Figure 9.2- 11: Truth element for Exploitation support data

ES.WFClp

- + ChNum.*TileNum*(ipi,iray)
- + Locs.*TileNum*(ipi,iray,j)
- + RtnsCode.*TileNum*(iray,ipi)

Figure 9.2- 12: Waveform element for Exploitation support data

Table 3: ES.* --Exploitation Support

Table Line #	Tag Name	Brief Definition	Domain	Essential/ not
	ES.*	Exploitation Support class of data	1-168	
	ES.Align.*	"Align" property of ES Items associated with the alignment of the geospatial data.	1-5	
1.	ES.Align.LocG_Vert(it,j)	gives the local gravitational vertical at the center of the geographical region for the tile indicated by j_tile.	**number**	
2.	ES.Align.X_ECEF(j_tile)	gives the orientation of the tile local Cartesian frame x-axis in the ECEF frame	**number**	
3.	ES.Align.Y_ECEF(j_tile)	gives the orientation of the tile local Cartesian frame y-axis in the ECEF frame	**number**	
4.	ES.Align.Z_ECEF(j_tile)	gives the orientation of the tile local Cartesian frame z-axis in the ECEF frame	**number**	
5.	ES.Align.XYZ_Flag	indicates whether the earth frame local (tile level) Cartesian frame of the data is aligned with a zone based, centroid based or ECEF based east-north-up X,Y, Z axis choice.	**code**	
	ES.CollCond.*	"CollCond" property of ES Items associated with the collection conditions	6-7	
6.	ES.CollCondn.Obscurant	is a flag indicating the nature of any obscurants affecting the data collection for the data set.	**code**	
7.	ES.CollCondn.PercentArea Affected	indicates the approximate percentage of ground area in sensor product view region voided due to obscuration	**number**	
	ES.Offset.*	"Offset" property of ES Items describing an additive term in the linear transform to be applied to point data	8-10	
8.	ES.Offset.X(j_tile)	carries the x- intercept term in a linear transformation to be applied to the lidar data in a tile to form actual	**number**	
9.	ES.Offset.Y(j_tile)	carries the y- intercept term in a linear transformation to be applied to the lidar data in a tile to form actual position data within the Cartesian frame of the tile within a product.	**number**	
10.	ES.Offset.Z(j_tile)	carries the z- intercept term in a linear transformation to be applied to the lidar data in a tile to form actual position data within the Cartesian frame of the tile within a product.	**number**	
	ES.Origin.*	"Origin" property of ES Items describing the origin of the local coordinate frame.	11-18	
11.	ES.Origin.GeodtcEpoch	gives the geodetic model name that determines the fine position of the Prime meridian plane that defines ECEF X=0. May impact ellipsoid parameters as well.	**string**	
12.	ES.Origin.MGRS_Flag	is a code that indicates the relationship between	**number**	

		the Local Cartesian origin choice, tiles and MGRS zones.		
13.	ES.Origin.X_ECEF(j_tile)	is the ECEF x-coordinate of the origin of the local Cartesian frame used by the j_tileth tile.	**number**	
14.	ES.Origin.Y_ECEF(j_tile)	is the ECEF y-coordinate of the origin of the local Cartesian frame used by the j_tileth tile.	**number**	
15.	ES.Origin.Z_ECEF(j_tile)	is the ECEF z-coordinate of the origin of the local Cartesian frame used by the j_tileth tile.	**number**	
16.	ES.Origin.X_MGRS(j_tile)	gives the easting value of the local Cartesian frame origin in the MGRS zone identified for the tile.	**number**	
17.	ES.Origin.Y_MGRS(j_tile)	gives the northing value of the local Cartesian frame origin in the MGRS zone identified for the tile.	**number**	
18.	ES.Origin.Z_MGRS(j_tile)	gives the height value of the local Cartesian frame origin in the MGRS zone identified for the tile.	**number**	
	ES.PDOP.*	"PDOP" property of ES Items associated with Position Dilution of Precision for multi-look product	19-21	
19.	ES.PDOP.Max	is the largest value of ES.PDOP.Val(j_tile) among all of the tiles in a data set.	**number**	
20.	ES.PDOP.Min	is the smallest (best) value of ES.PDOP.Val(j_tile) among all of the tiles in a data set.	**number**	
21.	ES.PDOP.Val(j_tile)	is the minimum Position Dilution of Precision for the point cloud data in a single tile.	**number**	
	ES.Point.*	"Point" property of ES Point level data	22-32	
22.	ES.Point.CharCode.*TileNum*(ipt)	ES.Point.CharCode.*TileNum*(ipt) carries a code characterizing a return	**code**	
23.	ES.Point.FilterScore.*TileNum*(ipt)	gives a filter score that may be used as a thresholdable display parameter to permit an analyst to vary the number of points within a tile to display based on the probability that each displayed point is signal rather than noise or clutter.	**number**	
24.	ES.Point.IntnsyFlg	indicates whether the measured intensities were in log (that is, dB) space before scaling and possible remapping to the stored values in the data set.	**code**	
25.	ES.Point.IntnsyRemap(j,k)	gives parameters for a piecewise linear remapping that has been applied to reduce the wordsize needed to convey the intensity data.	**number**	
26.	ES.Point.LM_SigmaR.*TileNum*(ipt)	is the linear mode range uncertainty on a point by point basis.	**number**	
27.	ES.Point.LM_SigmaT.*TileNum*(ipt)	is the linear mode round trip time uncertainty on a point by point basis.	**number**	
28.	ES.Point.LookSpread.*Tile*	is the angle between the look directions	**number**	

	Num(ipt)	between the points in a tile that have the greatest difference in line of sight.		
29.	ES.Point.PulseNum.*TileNum*(ipt,j)	provides the pulse number (also referred to as the pulse index) for which a specific point was generated.	**number**	
30.	ES.Point.RtnsCode.*TileNum*(ipt)	encodes information about the return that results in lidar point (ipt)	**code**	
31.	ES.Point.SigmaXYZ(j,k)	is a list of pre-computed point error ellipsoid parameter value sets that apply to specific ranges of points in the data set.	**number**	
32.	ES.Point.TruthTags.*TileNum*(ipt,j)	enables points to be labled as associated with known truthed locations vs merely lidar sensor system derived locations.	**number**	
	ES.SAL.*	parameters describe synthetic aperture output data, when applicable.	33-39	
33.	ES.SAL.GridAxes(j)	gives the directions of the row (i) -axis direction and cross-row (j)- axis direction in the ECEF frame of the output data grid, LD.SAL.DataGrid.*ChNum.TileNum*(i,j,k). Applies for Synthetic Aperture Lidar processing only.	**number**	
34.	ES.SAL.GridType	indicates range-Doppler vs along track-cross track output coordinates	**code**	
35.	ES.SAL.IPR(j)	Indicates the output plane IPR -3dB width	**number**	
36.	ES.SAL.ORP(j)	provides the postionof the output reference point in the complex SAL data grid	**number**	
37.	ES.SAL.SL_RgAxis(j)	indicates the output plane range sidelobe axis direction.	**number**	
38.	ES.SAL.SL_AzAxis(j)	indicates the output plane range sidelobe axis direction.	**number**	
39.	ES.SAL.StepSize(j)	provides the range like and cross range like sample step size in the SAL output data grid	**number**	
	ES.Scale.*	"Scale" property of ES Coefficients in linear transform to apply to point position data	40-42	
40.	ES.Scale.X_Fac	is the coefficient of the x axis data in the linear transformations that produce x positions in meters from stored x-components of lidar point position.	**number**	
41.	ES.Scale.Y_Fac	is the coefficient of the y axis data in the linear transformations that produce y positions in meters from stored y-components of lidar point position.	**number**	
42.	ES.Scale.Z_Fac	is the coefficient of the z axis data in the linear transformations that produce z positions in meters from stored z-components of lidar point position.	**number**	
	ES.Scan.*	"Scan" property of ES Scan level data items	43-110	

43.	ES.Scan.CorrParms.Alpha(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal x-reference direction at the sensor.	**number**	
44.	ES.Scan.CorrParms.Beta(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal y-reference direction at the sensor.	**number**	
45.	ES.Scan.CorrParms.Gamma(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the net rotation about the LOS at the sensor.	**number**	
46.	ES.Scan.CorrParms.X(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the net sensor position x-coordinate in the earth frame.	**number**	
47.	ES.Scan.CorrParms.Y(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the net sensor position y-coordinate in the earth frame.	**number**	
48.	ES.Scan.CorrParms.Z(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the net sensor position z-coordinate in the earth frame.	**number**	
49.	ES.Scan.CorrParms.R(j,n,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error in the range from sensor to a lidar point.	**number**	
50.	ES.Scan.CorrParms.AlphaBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and beta in the earth frame.	**number**	
51.	ES.Scan.CorrParms.AlphaGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and gamma in the earth frame.	**number**	
52.	ES.Scan.CorrParms.BetaGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error	**number**	

		correlation model as a function of time for the error cross covariance in the net sensor orientations beta and gamma in the earth frame.		
53.	ES.Scan.CorrParms.XY(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and y in the earth frame.	**number**	
54.	ES.Scan.CorrParms.XZ(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and z in the earth frame.	**number**	
55.	ES.Scan.CorrParms.XAlpha(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation alpha in the earth frame.	**number**	
56.	ES.Scan.CorrParms.XBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation beta in the earth frame.	**number**	
57.	ES.Scan.CorrParms.XGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation gamma in the earth frame.	**number**	
58.	ES.Scan.CorrParms.YZ(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinates y and z in the earth frame.	**number**	
59.	ES.Scan.CorrParms.YAlpha(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle alpha in the earth frame.	**number**	
60.	ES.Scan.CorrParms.YBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle beta in the earth frame.	**number**	

61.	ES.Scan.CorrParms.YGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle gamma in the earth frame.	**number**	
62.	ES.Scan.CorrParms.ZAlpha(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle alpha in the earth frame.	**number**	
63.	ES.Scan.CorrParms.ZBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle beta in the earth frame.	**number**	
64.	ES.Scan.CorrParms.ZGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning intra-snapshot error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle gamma in the earth frame.	**number**	
65.	ES.Scan.CorrParmsT.Alpha(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal x-reference direction at the sensor.	**number**	
66.	ES.Scan.CorrParmsT.Beta(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal y-reference direction at the sensor.	**number**	
67.	ES.Scan.CorrParmsT.Gamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net rotation about the LOS at the sensor.	**number**	
68.	ES.Scan.CorrParmsT.X(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net sensor position x-coordinate in the earth frame.	**number**	
69.	ES.Scan.CorrParmsT.Y(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the	**number**	

		error in the net sensor position y-coordinate in the earth frame.		
70.	ES.Scan.CorrParmsT.Z(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net sensor position z-coordinate in the earth frame.	**number**	
71.	ES.Scan.CorrParmsT.R(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the range from sensor to a lidar point.	**number**	
72.	ES.Scan.CorrParmsT.AlphaBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning inter-snapshot error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and beta in the earth frame.	**number**	
73.	ES.Scan.CorrParmsT.AlphaGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and gamma in the earth frame.	**number**	
74.	ES.Scan.CorrParmsT.BetaGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations beta and gamma in the earth frame.	**number**	
75.	ES.Scan.CorrParmsT.XY(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and y in the earth frame.	**number**	
76.	ES.Scan.CorrParmsT.XZ(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and z in the earth frame.	**number**	
77.	ES.Scan.CorrParmsT.XAlpha(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation alpha in the earth frame.	**number**	
78.	ES.Scan.CorrParmsT.XBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation beta in the earth	**number**	

		frame.		
79.	ES.Scan.CorrParmsT.XGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation gamma in the earth frame.	**number**	
80.	ES.Scan.CorrParmsT.YZ(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates y and z in the earth frame.	**number**	
81.	ES.Scan.CorrParmsT.YAlpha(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle alpha in the earth frame.	**number**	
82.	ES.Scan.CorrParmsT.YBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle beta in the earth frame.	**number**	
83.	ES.Scan.CorrParmsT.YGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle gamma in the earth frame.	**number**	
84.	ES.Scan.CorrParmsT.ZAlpha(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle alpha in the earth frame.	**number**	
85.	ES.Scan.CorrParmsT.ZBeta(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle beta in the earth frame.	**number**	
86.	ES.Scan.CorrParmsT.ZGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle gamma in the earth frame.	**number**	
87.	ES.Scan.DelT_Cov	is the time increment between scan level error	**number**	

		covariance data "snapshots"		
88.	ES.Scan.GM_SigmaR(j)	is the range uncertainty at the scan level. (Geiger[15]Mode only)	**number**	
89.	ES.Scan.RhoAlphaGamma j)	is the scan level correlation between the uncertainty in cos(alpha) and the uncertainty in the angle gamma, and is part of the sensor error covariance data.	**number**	
90.	ES.Scan.RhoXBeta(j)	is the scan level correlation between the x-position coordinate and angle beta, and is part of the sensor error covariance data.	**number**	
91.	ES.Scan.RhoAlphaBeta(j)	is the scan level correlation between uncertainties in the cosines of the line of sight angles alpha and beta, and is part of the sensor error covariance data.	**number**	
92.	ES.Scan.RhoBetaGamma(j)	is the scan level correlation between the uncertainty in cos(beta) and the uncertainty in the angle gamma, and is part of the sensor error covariance data.	**number**	
93.	ES.Scan.RhoXAlpha(j)	is the scan level correlation between the x-position coordinate and cosine of angle alpha, and is part of the sensor error covariance data.	**number**	
94.	ES.Scan.RhoXGamma(j)	ES.Scan.RhoXGamma(j) is the scan level correlation between the sensor x-position coordinate uncertainty and the uncertainty in the angle gamma, and is part of the sensor error covariance data.	**number**	
95.	ES.Scan.RhoXY(j)	is the scan level correlation between the uncertainty in the sensor x-position coordinate and uncertainty in the y-position coordinate, and is part of the sensor error covariance data.	**number**	
96.	ES.Scan.RhoXZ(j)	is the scan level correlation between the uncertainty in the sensor x-position coordinate and the uncertainty in the z-position coordinate, and is part of the sensor error covariance data.	**number**	
97.	ES.Scan.RhoYAlpha(j)	is the scan level correlation between the y-position coordinate uncertainty and uncertainty in the cosine of angle alpha, and is part of the sensor error covariance data.	**number**	
98.	ES.Scan.RhoYBeta(j)	is the scan level correlation between the uncertainty in the y-position coordinate and	**number**	

[15] Alternative to defining range uncertainty for Geiger mode only is to have generic sigmaR with mode dependent statements in the definition. For this draft, we have not consolidated so as to simplify data interpretation for possible level-4 products that might combine a linear mode and a Geiger mode point cloud. There are other level-4 issues with such a possibility and the idea of error models for level-4 has yet to be well framed, partly due to the range of possibilities in forming L-4 products. WE may ultimately get rid of the separate tags for GM and LM range uncertainties.

		uncertainty in the cosine of angle beta, and is part of the sensor error covariance data.		
99.	ES.Scan.RhoYGamma(j)	is the scan level correlation between the uncertainty in the sensor y-position coordinate and uncertainity in the angle gamma, and is part of the sensor error covariance data.	**number**	
100.	ES.Scan.RhoYZ(j)	is the scan level correlation between the uncertainty in the sensor y-position coordinate and uncertainty in the sensor z-position coordinate, and is part of the sensor error covariance data.	**number**	
101.	ES.Scan.RhoZAlpha(j)	is the scan level correlation between the uncertainty in the sensor z-position coordinate and the uncertainty in the cosine of angle alpha, and is part of the sensor covariance data.	**number**	
102.	ES.Scan.RhoZBeta(j)	is the scan level correlation between uncertainty in the z-position coordinate and uncertainty in the cosine of angle beta, and is part of the sensor error covariance data.	**number**	
103.	ES.Scan.RhoZGamma(j)	is the scan level correlation between the uncertainity in the sensor z-position coordinate and uncertainty in the angle, gamma, and is part of the sensor error covariance data.	**number**	
104.	ES.Scan.SigmaAlpha(j)	describes the sensor orientation uncertainty and is given by the standard deviation of the cosine of angle alpha measurement, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
105.	ES.Scan.SigmaBeta(j)	describes the sensor orientation uncertainty and is given by the standard deviation of the cosine of angle beta measurement, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
106.	ES.Scan.SigmaGamma(j)	describes the sensor orientation uncertainty and is given by the standard deviation of the angle gamma measurement, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
107.	ES.Scan.SigmaX(j)	describes the sensor position uncertainty and is given by the standard deviation of the x-coordinate measurement, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
108.	ES.Scan.SigmaY(j)	describes the sensor position uncertainty and is given by the standard deviation of the y-coordinate measurement, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
109.	ES.Scan.SigmaZ(j)	describes the sensor position uncertainty and is	**number**	

		given by the standard deviation of the z-coordinate measurement, a contributor to the diagonal of the error covariance matrix for the sensor.		
110.	ES.Scan.SigmaNbar(j)	describes the uncertainty in the path averaged index of refraction.		
	ES.Swath.*	"Swath" property of ES Swath level data items	111-159	
111.	ES.Swath.CorrParms.Alpha(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal x-reference direction at the sensor.	**number**	
112.	ES.Swath.CorrParms.Beta(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal y-reference direction at the sensor.	**number**	
113.	ES.Swath.CorrParms.Gamma(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net rotation about the LOS at the sensor.	**number**	
114.	ES.Swath.CorrParms.X(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net sensor position x-coordinate in the earth frame.	**number**	
115.	ES.Swath.CorrParms.Y(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net sensor position y-coordinate in the earth frame.	**number**	
116.	ES.Swath.CorrParms.Z(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net sensor position z-coordinate in the earth frame.	**number**	
117.	ES.Swath.CorrParms.R(j,n,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the range measurement from the sensor to an output (likely synthetic) lidar point.	**number**	
118.	ES.Swath.CorrParms.AlphaBeta(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and beta in the earth frame.	**number**	

119.	ES.Swath.CorrParms.AlphaGamma(j,k)	gives the values of the parameters of the pre-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and gamma in the earth frame.	**number**	
120.	ES.Swath.CorrParms.BetaGamma(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor orientations beta and gamma in the earth frame.	**number**	
121.	ES.Scan.CorrParms.XY(j,k)	gives the values of the parameters of the pre-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and y in the earth frame.	**number**	
122.	ES.Swath.CorrParms.XZ(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and z in the earth frame.	**number**	
123.	ES.Swath.CorrParms.XAlpha(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation alpha in the earth frame.	**number**	
124.	ES.Swath.CorrParms.XBeta(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation beta in the earth frame.	**number**	
125.	ES.Swath.CorrParms.XGamma(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation gamma in the earth frame.	**number**	
126.	ES.Swath.CorrParms.YZ(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinates y and z in the earth frame.	**number**	
127.	ES.Swath.CorrParms.YAlpha(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle alpha in the earth frame.	**number**	
128.	ES.Swath.CorrParms.YBeta(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross	**number**	

		covariance in the net sensor position coordinate y and orientation angle beta in the earth frame.		
129.	ES.Swath.CorrParms.YGamma(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle gamma in the earth frame.	**number**	
130.	ES.Swath.CorrParms.ZAlpha(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle alpha in the earth frame.	**number**	
131.	ES.Swath.CorrParms.ZBeta(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle beta in the earth frame.	**number**	
132.	ES.Swath.CorrParms.ZGamma(j,k)	gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle gamma in the earth frame.	**number**	
133.	ES.Swath.DelT_Cov	is the time increment between swath level covariance data "snapshots"	**number**	
134.	ES.Swath.GM_SigmaRfrac(i,j)	provides the fraction of the raw point range uncertainty remaining after processing to reduce the impact of a range point spread due to sampling the range at intervals finer than transmit pulse length.	**number**	
135.	ES.Swath.GM_SigmaT	is the uncertainty in roundtrip timing for Geiger Mode for a given data set.	**number**	
136.	ES.Swath.RhoAlphaGamma(k)	is the post processing correlation between the uncertainties in cos(alpha) and the angle gamma, part of the sensor error covariance data.	**number**	
137.	ES.Swath.RhoXBeta(k)	is the post processing correlation between the uncertainties in the x-position coordinate and the cosine of the angle beta, and is part of the sensor error covariance data.	**number**	
138.	ES.Swath.RhoAlphaBeta(k)	is the post-processing correlation between the uncertainties in the cosines of the line of sight angles, alpha and beta, and is part of the sensor error covariance data.	**number**	
139.	ES.Swath.RhoBetaGamma(k)	is the post-processing correlation between the uncertainties in cos(beta) and the angle,gamma, part of the sensor error covariance data.	**number**	
140.	ES.Swath.RhoXAlpha(k)	is the post-processing correlation between the	**number**	

		uncertainties in the x-position coordinate and cosine of angle alpha, and is part of the sensor covariance data.		
141.	ES.Swath.RhoXGamma(k)	is the post-processing correlation between uncertainties in the sensor x-position coordinate and the angle, gamma, and is part of the sensor error covariance data.	**number**	
142.	ES.Swath.RhoXY(k)	is the post-processing correlation between uncertainties in the sensor x-position coordinate and y-position coordinate, and is part of the sensor error covariance data.	**number**	
143.	ES.Swath.RhoXZ(k)	is the post-processing correlation between uncertainties in the sensor x-position coordinate and z-position coordinate, and is part of the sensor error covariance data.	**number**	
144.	ES.Swath.RhoYAlpha(k)	is the post processing correlation between uncertainties in the y-position coordinate and cosine of angle alpha, and is part of the sensor error covariance data.	**number**	
145.	ES.Swath.RhoYBeta(k)	is the post processsing correlation between uncertainties in the y-position coordinate and cosine of angle beta, and is part of the sensor error covariance data.	**number**	
146.	ES.Swath.RhoYGamma(k)	is the post processing correlation between uncertainties in the sensor y-position coordinate and the angle, gamma, and is part of the sensor error covariance data.	**number**	
147.	ES.Swath.RhoYZ(k)	is the post processing correlation between uncertainties in the sensor y-position coordinate and z-position coordinate, and is part of the sensor error covariance data.	**number**	
148.	ES.Swath.RhoZAlpha(k)	is the post-processing correlation between uncertainties in the sensor z-position coordinate and the cosine of the angle alpha, and is part of the sensor error covariance data.	**number**	
149.	ES.Swath.RhoZBeta(k)	is the post-processing correlation between uncertainties in the z-position coordinate and cosine of angle beta, and is part of the sensor error covariance data.	**number**	
150.	ES.Swath.RhoZGamma(k)	is the post-processing correlation between uncertainties in the sensor z-position coordinate and angle, gamma, and is part of the sensor error covariance data.	**number**	
151.	ES.Swath.SigmaAlpha(k)	is the post-processing standard deviation of the cosine of angle alpha, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
152.	ES.Swath.SigmaBeta(k)	is the post-processing standard deviation of the measurement of cosine of angle beta, a	**number**	

		contributor to the diagonal of the error covariance matrix for the sensor.		
153.	ES.Swath.SigmaGamma(k)	is the post-processing standard deviation of the measurement of angle gamma, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
154.	ES.Swath.SigmaX(k)	is the post processing standard deviation of the measurement of the x-coordinate of sensor position, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
155.	ES.Swath.SigmaY(k)	is the post-processing standard deviation of the measurement of the y-coordinate of sensor position, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
156.	ES.Swath.SigmaZ(k)	is the post-processing standard deviation of the measurement of the z-coordinate of sensor position, a contributor to the diagonal of the error covariance matrix for the sensor.	**number**	
157.	ES.Swath.SensorBasedCoordFlg(j)	indicates the coordinate system and frame orientation for point positions presented in the sensor frame.	**code**	
158.	ES.Swath.HiConfdncP_Fac(i,j,k)	multiplied by a standard deviation uncertainty (such as ES.Swath.SigmaT) provides the high confidence error bars of the quantity whose uncertainty is being provided.	**number**	
159.	ES.Swath.PSF(j)	indicates the spatial size of a focal plane element's angular resolution cell at the nominal range of the ground for a collection.	**number**	
	ES.Truth.*		160-165	
160.	ES.Truth.CalChRatios(ich)	provides the factors that convert individual channel intensity data into relative radiometric channel intensity data.	**number**	
161.	ES.Truth.IntnsyRef(itp,ich)	provides spectral band and polarization specific reflectivity for truthed points in the scene of the lidar data set.	**number**	
162.	ES.Truth.IllusKMZ.*j*(nchar)	provides an illustration of the local region about ground truth points in the lidar data set.	**Char string**	
163.	ES.Truth.Pos(itp,j)	provides geodetic coordinates for truthed points in the scene of the lidar data set.	**number**	
164.	ES.Truth.CovCorrs(itp,j)	provides the position error correlations for the positions of the truth points provided in ES.Truth.Pos(itp,j).	**number**	
165.	ES.Truth.CovSigs(itp,j)	provides the error standard deviations for the positions of the truth points provided in ES.Truth.Pos(itp,j).	**number**	
	ES.WFClp.*	Waveform clip related	166-168	
166.	ES.WFClp.ChNum.*TileNum*(ipi, iray)	provides channel identification for waveform data.	**Char string**	

167.	ES.WFClp.Locs.*TileNum*(ipi,iray,j)	provides location information about the return associated with the waveform clip identified by pulse, ipi, and ray, iray.	**number**	
168.	ES.WFClp.RtnsCode.*TileNum*(ip,iray)	encodes information about the return	**code**	

9.3. PF –Product Formation

Product formation metadata provides information needed in further processing and interpreting the lidar data. It explains how the data in the product came to have its current characteristics.

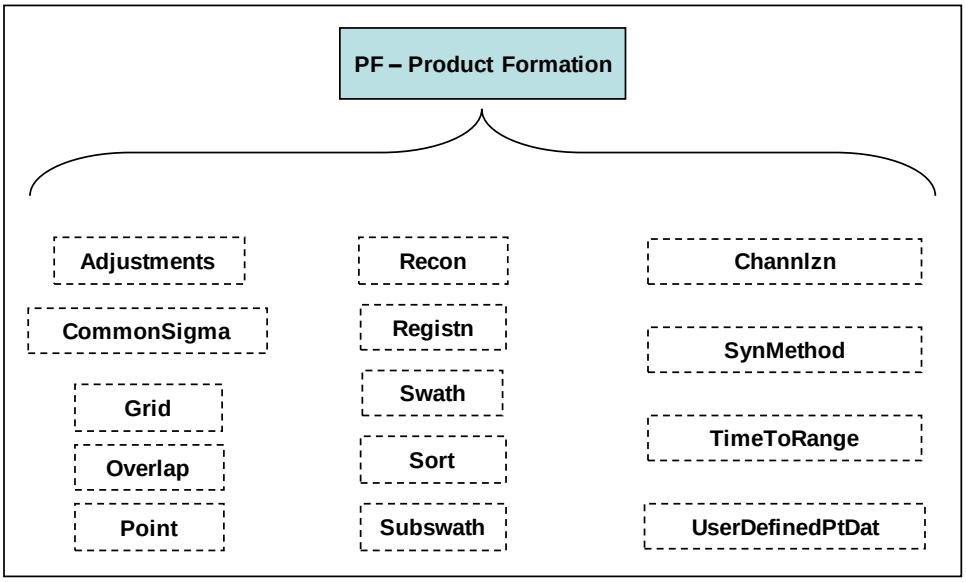


Figure 9.3- 1: PF Product Formation second level data structure

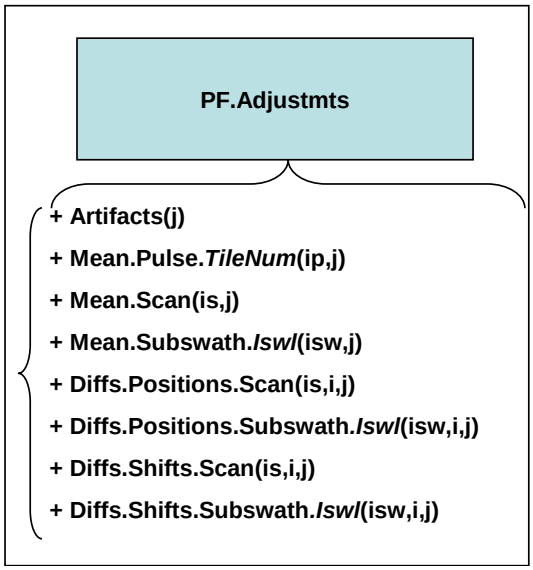


Figure 9.3- 2: Adjustments elements for Product Formation

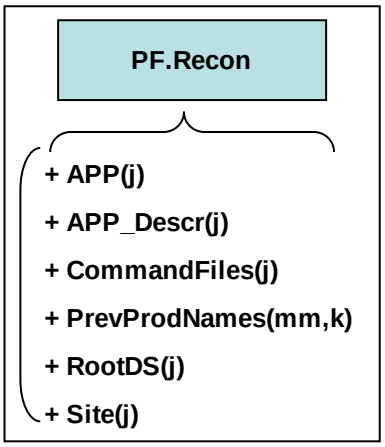


Figure 9.3- 3: Reconstruction elements for Product Formation

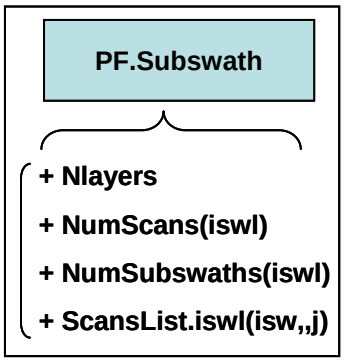


Figure 9.3- 4: Subswath elements for Product Formation

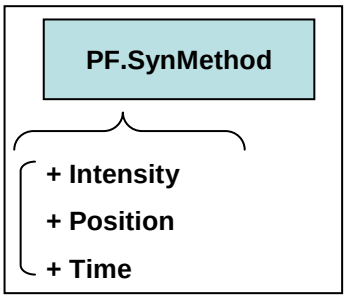


Figure 9.3- 5: SynMethod elements for Product Formation

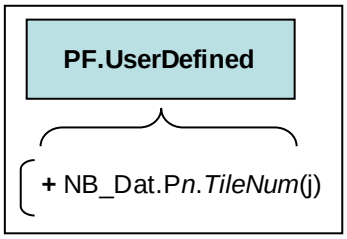


Figure 9.3- 6: User defined elements for Product Formation

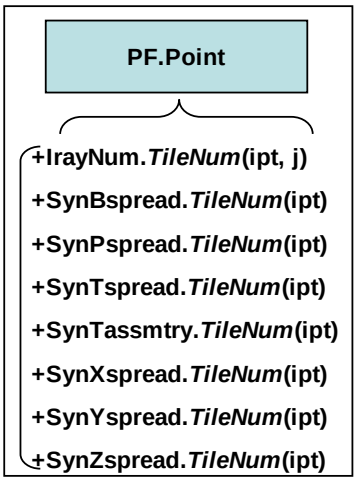


Figure 9.3- 7: Point elements for Product Formation

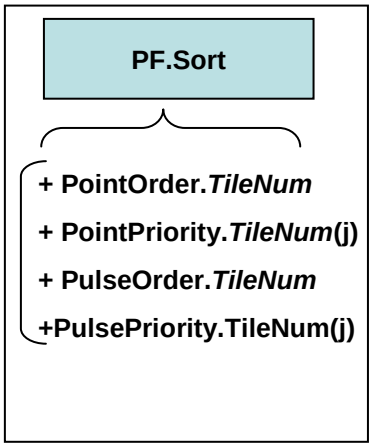


Figure 9.3- 8: Sort elements for Product Formation

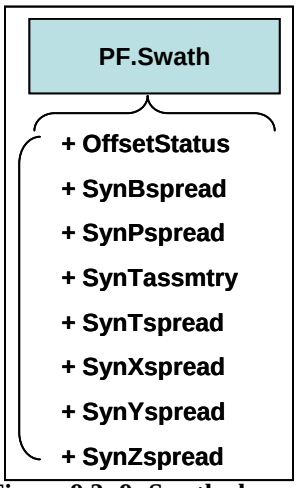


Figure 9.3- 9: Swath elements for Product Formation

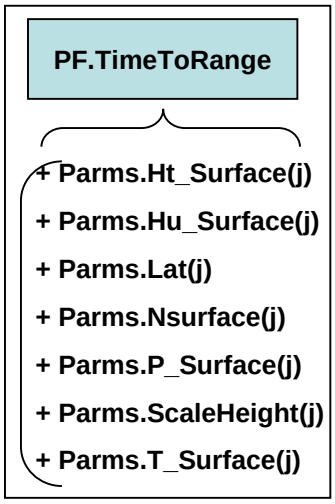


Figure 9.3- 10: Time to Range elements for Product Formation

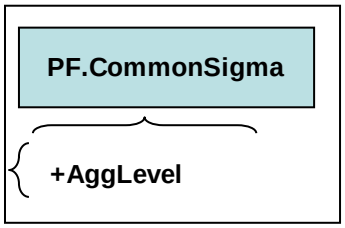


Figure 9.3- 11: Common Sigma elements for Product Formation

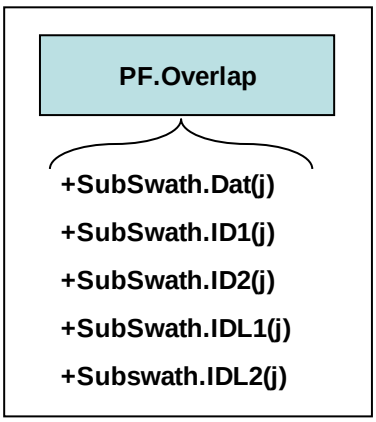


Figure 9.3- 12: Overlap elements for Product Formation

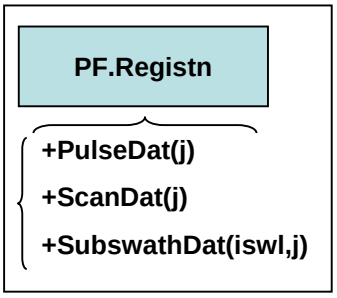


Figure 9.3- 14: Registration elements for Product Formation

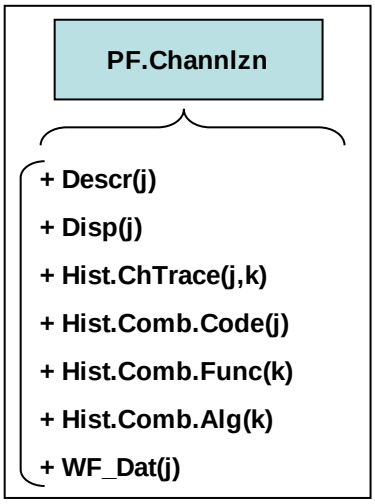


Figure 9.3- 13: Channelization on elements for Product Formation

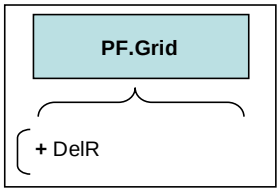


Figure 9.3- 15: Grid element for Product Formation

Table 4: PF.* --Product Formation

Table Line #	Tag Name	Brief Definition	Domain	Essential/ not
	PF.*	Product Formation class of data	1-77	
	PF.Adjustmts	Items associated with registration derived coordinate corrections	1-8	
1.	PF.Adjustmts.Artifacts(j)	provides indication of the effectiveness of remedies for known data artifacts	**code**	
2.	PF.Adjustmts.Mean.Pulse.*TileNum*(ip,j)	provides the aligning transformation between frames when frame to frame registration is performed in the construction of scans.	**number**	
3.	PF.Adjustmts.Mean.Scan(is,j)	provides the aligning coordinate transformation between scan pairs in the creation of subswaths.	**number**	
4.	PF.Adjustmts.Mean.Subswath.*Iswl*(isw,j)	provides the aligning coordinate transformation between subswath pairs in the creation of larger subswaths or the final swath.	**number**	

5.	PF.Adjustmts.Diffs.Positions.Scan(is, i, j)	provides the earth frame spatial positions to which demeaned shifts for an aligning transformation at the scan level in the creation of subswaths are attributed.	**number**	
6.	PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j)	provides the earth frame spatial positions to which demeaned shifts for an aligning transformation at thew subswath level in the creation of a larger aggregation of points are attributed.	**number**	
7.	PF.Adjustmts.Diffs.Shifts.Scan(is,i, j)	provides the shifts for the aligning transformation at the scan level in the creation of subswaths.	**number**	
8.	PF.Adjustmts.Diffs.Shifts.Subswath.*Iswl*(isw,i, j)	provides the aligning coordinate transformation at subswath layerl *Iswl* in the creation of larger subswaths or the final swath.	**number**	
	PF.Channlzn	"Channlzn" (Channelization) property of PF. Items associated with channels due to processing of channels in parent data set.	9-15	
9.	PF.Channlzn.Descr(j)	provides a description for each channel of data identified in a lidar data set.	**Char string**	
10.	PF.Channlzn.Disp(j)	documents the intended use of channels in a lidar data set	**code**	
11.	PF.Channlzn.Hist.ChTrace(j,k)	indicates the parent channels for each channel in a L-2 or higher lidar data set.	**code**	
12.	PF.Channlzn.Hist.Comb.Code(j)	documents the combination methodology at top level.	**code**	
13.	PF.Channlzn.Hist.Comb.Func(k)	provides second level detail for channels created using simple equations other than RSS and Linear combination.	**Char string**	
14.	PF.Channlzn.Hist.Comb.Alg(k)	provides the registerd name for complex channel combination algorithms applied in creating the channels within a lidar data set.	**Char string**	
15.	PF.Channlzn.Hist.WF_Dat(j)	provides parameters for combining waveform data to create combined channels at the waveform level within a lidar data set.	**number**	
	PF.CommonSigma.*	"CommonSigma" property of PF Items associated with common uncertainty values for the entire logical or spatial levels within the data set.	16-16	
16.	PF.CommonSigma.AggLevel	code: 1-pulse level sigmaXYZ reported 2-scan level sigmaXYZ reported 3-subswath level sigmaXYZ reported 4-tile level sigmaXYZ reported 5-swath level sigmaXYZ reported	**code**	
	PF.Grid.*	"Grid" property of PD Items associated with an explicit or implicit gridding of the point data.	17-17	
17.	PF.Grid.DelR	is the range increment for intermediate data	**number**	

		given on a range grid to create the current data set, if used. **Currently renamed from PD to PF.*. candidate for deletion.**		
	PF.Overlap.*	"Overlap" property of PF Items associated with overlaps between data set geospatial components.	18-22	
18.	PF.Overlap.SubSwath.Dat(j)	provides the nominal percentage of subswath area in processing generated overlaps	**number**	
19.	PF.Overlap.SubSwath.ID1(j)	ID1 is isw of first subswath of pair j= 1 to number of subswath pairs.	**number**	
20.	PF.Overlap.SubSwath.ID2(j)	ID2 isw of second subswath of pair j= 1 to number of subswath pairs.	**number**	
21.	PF.Overlap.SubSwath.IDL1(j)	For processing subswaths: IDL1 is the iswl of first subswath of pair j= 1 to number of subswath pairs.	**number**	
22.	PF.Overlap.SubSwath.IDL2(j)	IDL2 is iswl of second subswath of pair j= 1 to number of subswath pairs.	**number**	
	PF.Point.*	"Point" property of PF Point level data items.	23-31	
23.	PF.Point.IrayNum.*TileNum*(ip,j)	identifies the focal plane detector array that a lidar point was collected with.	**number**	
24.	PF.Point.SenFrmRefPulse(j_tile)	provides the IPP number that indicates the **pulse** index value for the **ref**erence sensor position used for the j_tileth tile when computing **sen**sor **fra**me point positions.		
25.	PF.Point.SynBspread.*TileNum*(ipt)	indicates the point level degree of spectral spread represented by a synthetic lidar point.	**number**	
26.	PF.Point.SynPspread.*TileNum*(ipt)	indicates the point level degree of polarization spread represented by a synthetic lidar point.	**number**	
27.	PF.Point.SynTasymtry.*TileNum*(ipt)	is the time from the start of the interval supporting a synthetic point to the time of the IPP to which the synthetic point is assigned.	**number**	
28.	PF.Point.SynTspread.*TileNum*(ipt)	gives the point level duration of the interval contributing to the creation of a synthetic lidar point.	**number**	
29.	PF.Point.SynXspread.*TileNum*(ipt)	gives the point level x-direction spread in lidar points contributing to a single synthetic lidar point.	**number**	
30.	PF.Point.SynYspread.*TileNum*(i	gives the point level y-direction spread	**number**	

	pt)	in lidar points contributing to a single synthetic lidar point.		
31.	PF.Point.SynZspread.*TileNum*(ipt)	gives the point level z-direction spread in lidar points contributing to a single synthetic lidar point.	**number**	
	PF.Recon.*	"Recon" property of PF Items that would permit reconstruction of the current file WB data from precursor data.	32-37	
32.	PF.Recon.APP(j)	Each element is a unique site based identifier indicating what application was used in the processing chain that created the current data set from raw data.	**Char string**	
33.	PF.Recon.APP_Descr(j)	bears text descriptions of the applications in the processing chain leading to the current data set. Each description is a brief characterization of the key processing function performed by the APP	**Char string**	
34.	PF.Recon.PrevProdNames(mm,k)	provides the data set names for all precursor data sets in the processing chain for the kth parent data set of the current data set.	**Char string**	
35.	PF.Recon.RootDS(j)	provides the name and an archive location for the root wide band data set(s) processed to create a level-2 through 4 product.	**Char string**	
36.	PF.Recon.CommandFiles(j)	provides the values of input parameters for each of the application executions that led to creation of the current data set.	**Char string**	
37.	PF.Recon.Site(j)	provides the names of sites where applications were executed in the processing chain leading to the current data set.	**Char string**	
	PF.Registn	"Registration" property of PF Codes for processing treatment of collection overlaps	38-40	
38.	PF.Registn.PulseDat(j)	provides codes for the status of pulse to pulse registrations, if performed, to composite frames into scans.	**code**	
39.	PF.Registn.ScanDat(j)	provides codes for the status of scan to scan registrations, if performed, to composite scans into subswaths.	**code**	
40.	PF.Registn.SubswathDat(iswl,j)	provides codes for the status of scan group to scan group registrations, if performed, to composite subswaths into larger subswaths or the final product swath.	**code**	
	PF.SAL.*	parameters describe items related to synthetic aperture processing of the time domain lidar data.	41-48	
41.	PF.SAL.Autofocus	indicates the applied autofocus correction degree		
42.	PF.SAL.CollMode	indicates the synthetic aperture collection mode		

43.	PF.SAL.DOF	is the depth of focus in meters for the synthetic aperture collection.		
44.	PF.SAL.ProcGrid.Axes.*ChNum.TileNum*(j)	indicates the processing plane orientation with a unit vector for the range-like direction		
45.	PF.SAL.ProcGrid.Code	indicates whether the processing plane is a slant plane or ground plane.		
46.	PF.SAL.ProcGrid.Normal *ChNum.TileNum*(j)	indicates the processing plane orientation with an ECEF unit vector		
47.	PF.SAL.ProcGrid.GRP.*ChNum.TileNum*(j)	indicates the ECEF position of the ground reference profile used in processing the time domain lidar data		
48.	PF.SAL.ProcGrid.Types	carries a code to indicate the nature of the processing grid		
	PF.Sort.*	"Sort" property of PF Items documenting the sorting of data in the data set tiles.	49-52	
49.	PF.Sort.PointOrder.*TileNum*	ascend or descend flag value	**code**	
50.	PF.Sort.PointPriority.*TileNum*(j)	PF.Sort.PointPriority.*TileNum*(j) provides the priorities by which point data are sorted within a given tile.	**code**	
51.	PF.Sort.PulseOrder.*TileNum*	PF.Sort.PulseOrder.*TileNum* is a flag indicating whether pulse oriented data in the tile indicated by *TileNum* are sorted in increasing or decreasing order of the identified sorting parameters	**code**	
52.	PF.Sort.PulsePriority.*TileNum*(j)	PF.Sort.PulsePriority.*TileNum*(j) provides the priorities by which pulse data are sorted within a given tile.	**code**	
	PF.Subswath.*	"Subswath" property of PF Subswath level data items.	53-56	
53.	PF.Subswath.Nlayers	is the number of subswath layers chosen by the processor to create the current product.	**number**	
54.	PF.Subswath.NumScans(iswl)	gives the number of scans per subswath for each subswath layer.	**number**	
55.	PF.Subswath.Numswaths(iswl)	gives the number of subswaths for each subswath layer.	**number**	
56.	PF.Subswath.ScansList.*iswl*(isw, j)	identifies which scans are included in a particular subswath as specified by a subswath layer index (iswl) and a subswath index for that layer (isw).	**number**	
	PF.Swath.*	"Swath" property of PF Swath level data items.	57-64	
57.	PF.Swath.OffsetStatus	Flag to indicate whether supplied/applied offset and rotation data has been corrected with multilook or other added information vice being single look, original nav, based.	**code**	
58.	PF.Swath.SynBspread	is a characteristic value of the spectral bandwidth of synthetic points for the swath.	**number**	

59.	PF.Swath.SynPspread	is a characteristic value of the polarization variation contributing to synthetic points for the swath.	**number**	
60.	PF.Swath.SynTasymtry	is a characteristic value of the time asymmetry of synthetic points for the swath.	**number**	
61.	PF.Swath.SynTspread	is a characteristic value of the contributing time interval of synthetic points for the swath.	**number**	
62.	PF.Swath.SynXspread	is a characteristic value of the x-direction support region of synthetic points for the swath.	**number**	
63.	PF.Swath.SynYspread	is a characteristic value of the y-direction support region of synthetic points for the swath.	**number**	
64.	PF.Swath.SynZspread	is a characteristic value of the z-direction support region of synthetic points for the swath.	**number**	
	PF.SynMethod.*	"SynMethod" property of PF Items associated with the creation of synthetic lidar points.	65-67	
65.	PF.SynMethod.Intensity	is a code describing the method of computing the intensity for synthetic lidar points in the data set.	**code**	
66.	PF.SynMethod.Position	is a code describing the method of computing the position for synthetic lidar points in the data set.	**code**	
67.	PF.SynMethod.Time	supplies a code to describe how time is determined for synthetic lidar points in the current data set.	**code**	
	PF.TimeToRange.*	"TimeToRange" property of PF Items associated with converting photon time of flight to range.	68-76	
68.	PF.TimeToRange.BendEffect	is a code for what refractive impacts have been included in XYZ calculation.	**code**	
69.	PF.TimeToRange.Model	carries a code for the model used for converting photon time of flight to range.	**code**	
70.	PF.TimeToRange.Parms.Ht_Surface(j)	gives the value of scene base height to support atmospheric model surface gravity determination in the time to range conversion model.	**number**	
71.	PF.TimeToRange.Parms.Hu_Surface(j)	gives the surface value of humidity in terms of water partial pressure.	**number**	
72.	PF.TimeToRange.Parms.Lat(j)	gives the latitude of the scene for the current data set that was used in the refraction calculation.	**number**	
73.	PF.TimeToRange.Parms.Nsurface(j)	provides the index of refraction at the scene surface.	**number**	
74.	PF.TimeToRange.Parms.P_Surface(j)	gives the surface atmospheric pressure used in the time to range conversion.	**number**	
75.	PF.TimeToRange.Parms.ScaleHeight(j)	carries an array of scale heights relevant to stratified atmospheric models that may be invoked to determine the time to range conversion.	**number**	

76.	PF.TimeToRange.Parms.T_Surface(j)	gives the surface temperature used in the atmospheric refraction model for time to range conversion.	**number**	
	PF.UserDefined.*	"UserDefined" property of PF Items associated with User defined data parameters, which can be beta tested within compliant files. When the data definitions become stable, they may be given new tags and added to this document if wide usage is anticipated	77-77	
77.	PF.UserDefined.NB_Dat.P*n.TileNum*(j)	PF.UserDefined.NB_Dat.P*n.TileNum*(j) carries narrow band user defined data at the tile level.	**number**	

9.4. PD –Product Definition

Product Definition metadata provides information needed when ingesting or writing a VRP lidar data set. It explains how the wide band data in the product is arranged and supports finding specific data of interest within a large data set.

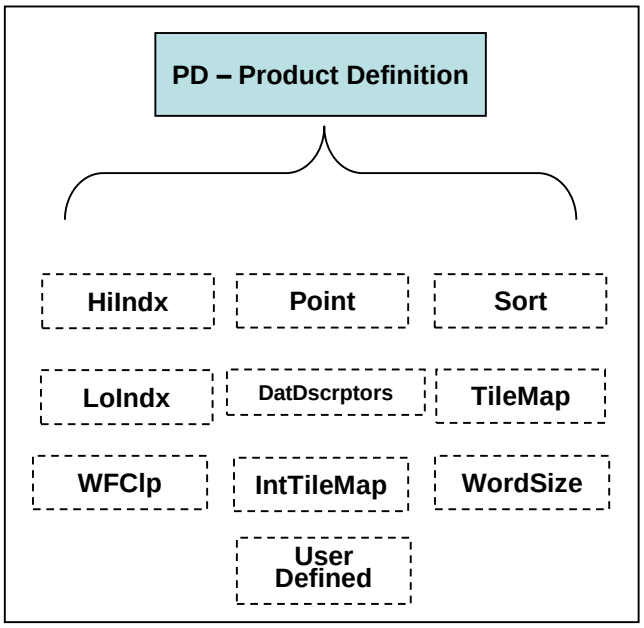


Figure 9.4- 1: Product Definition second level data

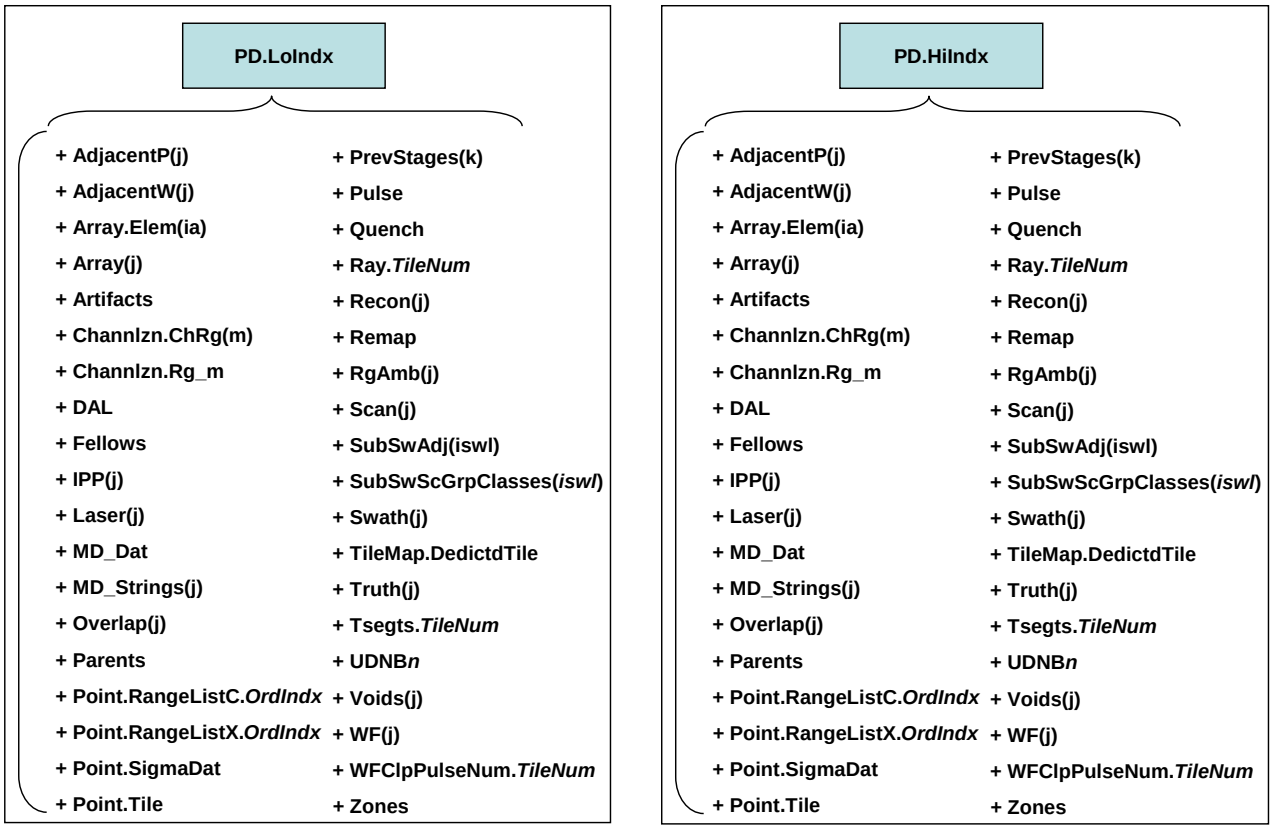


Figure 9.4- 2: LoIndx and HiIndx elements for Product Definition

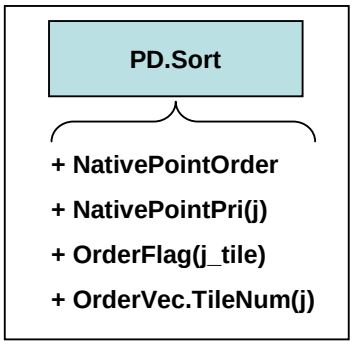


Figure 9.4- 3: Sort elements for Product Definition

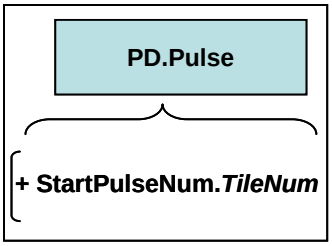


Figure 9.4- 4: Pulse elements for Product Definition

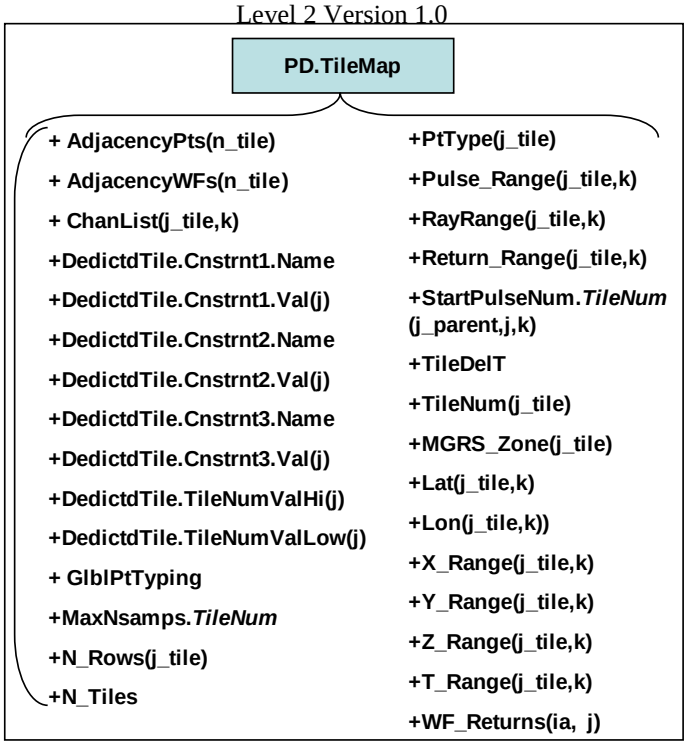


Figure 9.4- 5: TileMap elements for Product Definition

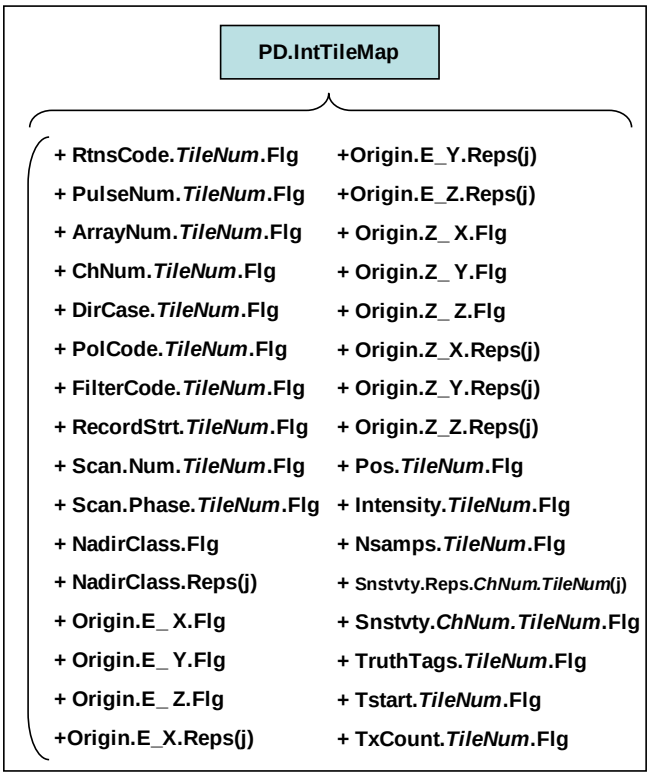


Figure 9.4- 6: Internal TileMap elements for Product Definition

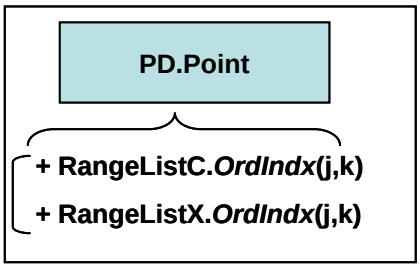


Figure 9.4- 7: Point elements for Product Definition

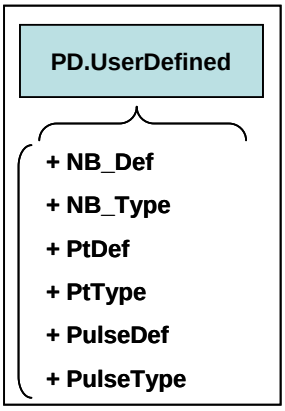


Figure 9.4- 8: User Defined elements for Product Definition

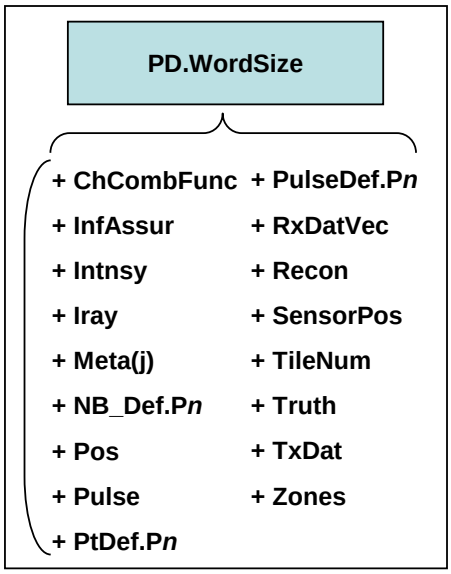


Figure 9.4- 9: Word Size elements for Product Definition

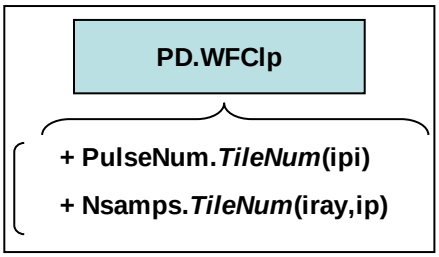


Figure 9.4- 10: WFClp elements for Product Definition

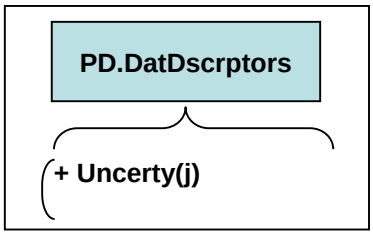


Figure 9.4- 11: DatDscrptors elements for Product Definition

Table 5: PD.* --Product Definition

Table Line #	Tag Name	Brief Definition	Domain	Essential/ not
	PD.*	Product Definition class of data	1-170	
	PD.DatDscrptors.*	data descripttors that specify values for generic conceptual data that need to be customizable for various implementations.		
1.	PD.DatDscrptors.Uncerty(j)	provides the percentile values for high confidence uncertainty statistics that can be generated using provided standard deviation and conversion factors.	**number**	C
	PD.LoIndx.*	"LoIndx" property of PD Items associated with starting indices for arrays in a lidar CMMD based data set. The "*" field uses character strings from the MD, ES, PF, SC families with arrays being described by PD Lo/HiIndx data. The spelling in the "*" field follows the rule: 'use as much name, but no more than, as needed to be unique'.	2-39	
2.	PD.LoIndx.AdjacentP(j)	provides the starting values for the TileMap point tile adjacency metadata.	**number**	
3.	PD.LoIndx.AdjacentW(j)	provides the starting values for the TileMap waveform tile adjacency metadata.	**number**	
4.	PD.LoIndx.Array(j)	gives the starting index values for data associated with the focal plane arrays	**number**	C
5.	PD.LoIndx.Array.Elem(ia,j)	is the starting value of the element index for the iath array.	**number**	C
6.	PD.LoIndx.Artifacts	gives the starting index values of SC.Artifacts.* and related items.	**number**	C
7.	PD.LoIndx.Channlzn.ChRg(m)	is the starting index for channels in channelization m. It supports the channel index ranges in channelization history data items such as PF.Channlzn.Hist.ChTrace.*m*(j,k) with minimum range values.	**number**	C
8.	PD.LoIndx.Channlzn.Rg_m	gives the minimum value of index, m, that appears in PD.HiIndx.Channlzn.ChRg(m) and PD.LoIndx.Channlzn.ChRg(m). This is	**number**	C

		the index associated with the original channelization presented to the PED.		
9.	PD.LoIndx.DAL	is the starting index for the list associated with the DAL version listdata sets of the current data set.	**number**	C
10.	PD.LoIndx.Fellows	is the starting index for lists associated with the fellow data sets of the current data set.	**number**	C
11.	PD.LoIndx.IPP(j)	supports the array index "ipn" ranges for SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) with minimum range values.	**number**	E
12.	PD.LoIndx.Laser(j)	supports the array index ranges for SC.Laser.* with minimum range values.	**number**	C
13.	PD.LoIndx.MD_Dat	gives the start of the "j" index range for MD.UseHist.* data items	**number**	C
14.	PD.LoIndx.MD_Strings(j)	gives start of the index range for PD.Wordsize.Meta(j)	**number**	E
15.	PD.LoIndx.Overlap(j)	supports the array index ranges for SC.Overlap.* and PF.Overlap.* with minimum range values.	**number**	C
16.	PD.LoIndx.Parents	is the starting index for lists associated with the parent data sets of the current data set.	**number**	C
17.	PD.LoIndx.Point.RangeListC.*OrdIndx*	is the starting value of the first index of PD.Point.RangeListC.*OrdIndx*(j,k)	**number**	C
18.	PD.LoIndx.Point.RangeListX.*OrdIndx*	is the starting value of the first index of PD.Point.RangeListX.*OrdIndx*(j,k)	**number**	C
19.	PD.LoIndx.Point.SigmaDat	is the starting index for ES.Point.SigmaXYZ(j,k) first index.	**number**	C
20.	PD.LoIndx.Point.Tile	gives the first value of the sequence of the j_tile index for tiles associated with point data.	**number**	E
21.	PD.LoIndx.PrevStages(k)	gives the first value of the index in PF.Recon.PrevProdNames(mm,k)	**number**	
22.	PD.LoIndx.Pulse	gives the first value of the sequence of the j_tile index for tiles associated with Pulse data.	**number**	C
23.	PD.LoIndx.Quench	provides the low value of the index for the array, SC.Quench.Times(j).	**number**	C
24.	PD.LoIndx.Ray.*TileNum*	gives the tile dependent starting value for index, iray, which covers the look directions for a given array.	**number**	C
25.	PD.LoIndx.Recon(j)	provides the low value of reconstruction data index ranges.	**number**	C
26.	PD.LoIndex.Remap	provides the low value of first index of ES.Point.IntnsyRemap(j,k)	**number**	C
27.	PD.LoIndx.RgAmb(j)	provides the low value of the indices for the array, SC.RgAmb.Num(i, j).	**number**	C
28.	PD.LoIndx.Scan(j)	provides start indices for scan level variable length arrays	**number**	C
29.	PD.LoIndx.SubSwAdj(iswl)	provides the starting index value for lists of	**number**	C

		adjustments at subswath level iswl.		
30.	PD.LoIndx.SubSwScGrpClasses(iswl)	supports the array index range of "it" in PF.SubSwath.ScansList.*iswl*(isw,it,j) with minimum range values.	**number**	C
31.	PD.LoIndx.Swath(j)	Provides start index values for swath level variable length arrays	**number**	C
32.	PD.LoIndx.TileMap.DedictdTile	gives the starting value for the index of dedicated tile description arrays.	**number**	C
33.	PD.LoIndx.Tsegts.*TileNum*	gives the first value of the range for the second index of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)	**number**	C
34.	PD.LoIndx.Truth(j)	gives the first value of indices in ground truth related data items.	**number**	C
35.	PD.LoIndx.UDNB*n*	gives the first value of the sequence of the j index for user defined narrow band data item PF.UserDefined.NB_Dat.P*n*.*TileNum*(j)	**number**	C
36.	PD.LoIndx.Voids(j)	gives the first value of the sequence of void index for each of the sequences of Void data in MD.Coverage.Voids.*	**number**	C
37.	PD.LoIndx.WF	gives the first value of the sequence of the j_tile index for tiles associated with waveform data.	**number**	C
38.	PD.LoIndx.WFClpPulseNum.*TileNum*	gives the first value of the index for the sequence of pulses used in a specific tile waveform clip data.	**number**	C
39.	PD.LoIndx.Zones	gives the first value of the lowest value of the index, j, of MD.Coverage.Bounds.ZoneList(j)	**number**	C
	PD.HiIndx.*	"HiIndx" property of PD Items associated with ending indices for arrays in a lidar CMMD based data set. The "*" field uses character strings from the MD, ES, PF, SC families with arrays being described by PD Lo/HiIndx data. The spelling in the "*" field follows the rule: 'use as much name, but no more than, as needed to be unique'.	40-77	
40.	PD.HiIndx.AdjacentP(j)	provides the ending values for the TileMap point tile adjacency metadata.	**number**	
41.	PD.HiIndx.AdjacentW(j)	provides the ending values for the TileMap waveform tile adjacency metadata.	**number**	
42.	PD.HiIndx.Array(j)	gives the ending index values for data associated with the focal plane arrays.	**number**	C
43.	PD.HiIndx.Array.Elem(ia,j)	is the ending value of the element index for the iath array.	**number**	C
44.	PD.HiIndx.Artifacts	gives the ending index values of SC.Artifacts.* and related items.	**number**	C
45.	PD.HiIndx.Channlzn.ChRg(m)	supports the array index ranges PF.Channlzn.Hist.ChTrace(j,k) with maximum range values.	**number**	C

46.	PD.HiIndx.Channlzn.Rg_m	gives the maximum value of index, m, that appears in PD.HiIndx.Channlzn.ChRg(m) and PD.LoIndx.Channlzn.ChRg(m). This is the channelization index corresponding to the current data set.	**number**	
47.	PD.HiIndx.DAL	is the ending index for the list associated with the DAL versions suitable for the current data set.	**number**	C
48.	PD.HiIndx.Fellows	is the ending index for lists associated with the fellow data sets of the current data set.	**number**	C
49.	PD.HiIndx.IPP(j)	supports the array index "ipn" ranges for SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) with maximum range values.	**number**	E
50.	PD.HiIndx.Laser(j)	supports the array index ranges for SC.Laser.* with maximum range values.	**number**	E
51.	PD.HiIndx.MD_Dat	gives the end of the "j" index range for MD.UseHist.* data items	**number**	C
52.	PD.HiIndx.MD_Strings(j)	gives end of the index range for PD.Wordsize.Meta(j)	**number**	E
53.	PD.HiIndx.Overlap(j)	supports the array index ranges for SC.Overlap.* with maximum range values.	**number**	C
54.	PD.HiIndx.Parents	is the ending index for lists associated with the parent data sets of the current data set.	**number**	C
55.	PD.HiIndx.Point.RangeListC.*OrdIndx*	is the ending value of the first index of PD.Point.RangeListC.*OrdIndx*(j,k)	**number**	C
56.	PD.HiIndx.Point.RangeListX.*OrdIndx*	is the ending value of the first index of PD.Point.RangeListX.*OrdIndx*(j,k)	**number**	C
57.	PD.HiIndx.Point.SigmaDat	is the ending index for ES.Point.SigmaXYZ(j,k) first index	**number**	C
58.	PD.HiIndx.Point.Tile	gives the last value of the sequence of the j_tile index for tiles associated with point data.	**number**	E
59.	PD.HiIndx.PrevStages(k)	gives the last value of the index, mm, in PF.Recon.PrevProdNames(mm,k)	**number**	
60.	PD.HiIndx.Pulse	gives the last value of the sequence of the j_tile index for tiles associated with Pulse data.	**number**	C
61.	PD.HiIndx.Quench	provides the high value of the index for the array, SC.Quench.Times(j).	**number**	C
62.	PD.HiIndx.Ray.*TileNum*	gives the tile dependent ending value for index, iray, which covers the look directions for a given array.	**number**	C
63.	PD.HiIndx.Recon(j)	provides the ending values of the reconstruction data index ranges.	**number**	C
64.	PD.HiIndx.Remap	provides the high value of first index of ES.Point.IntnsyRemap(j,k)	**number**	C
65.	PD.HiIndx.RgAmb(j)	determines the high value of the indices for the array, SC.RgAmb.Num.*Il*(isnc,j).	**number**	C

66.	PD.HiIndx.Scan(j)	Provides end indices for scan level variable length arrays	**number**	C
67.	PD.HiIndx.SubSwAdj(iswl)	provides the end index value for lists of adjustments at subswath level iswl.	**number**	C
68.	PD.HiIndx.SubSwScGrpClasses(iswl)	supports the array index range of "it" in PF.SubSwath.ScansList.*iswl*(isw,it,j) with maximum range values.	**number**	C
69.	PD.HiIndx.Swath(j)	Provides end index values for swath level variable length arrays	**number**	C
70.	PD.HiIndx.TileMap.DedictdTile	gives the ending value for the index of dedicated tile description arrays.	**number**	C
71.	PD.HiIndx.Tsegts.*TileNum*	gives the last value of the range for the second index of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)	**number**	C
72.	PD.HiIndx.Truth(j)	gives the last value of indices in ground truth related data items.	**number**	C
73.	PD.HiIndx.UDNB*n*	gives the last value of the sequence of the j index for user defined narrow band data item PF.UserDefined.NB_Dat.P*n*.*TileNum*(j)	**number**	C
74.	PD.HiIndx.Voids(j)	gives the last value of the void index for each of the sequences of Void data in MD.Coverage.Voids.*	**number**	C
75.	PD.HiIndx.WF	gives the last value of the sequence of the j_tile index for tiles associated with waveform data.	**number**	C
76.	PD.HiIndx.WFClpPulseNum.*TileNum*	gives the last value of the index for the sequence of pulses used in a specific tile waveform clip data.	**number**	C
77.	PD.HiIndx.Zones	gives the last value of the index, j, of MD.Coverage.Bounds.ZoneList(j)	**number**	C
	PD.Point	Point Level PD data	78-79	
78.	PD.Point.RangeListC.*OrdIndx*(j,k)	is a list of point index ranges for which the j1=*OrdIndx* error ellipsoid in ES.Point.SigmaXYZ(j1,k) is the default common value.	**number**	C
79.	PD.Point.RangeListX.*OrdIndx*(j,k)	Is a list of point index ranges for which the j1=*OrdIndx* error ellipsoid in ES.Point.SigmaXYZ.(j1,k) is the over riding exceptional value	**number**	C
	PD.Sort.*	"Sort" property of PD Items associated with the original sorting of the data within the file.	80-83	
80.	PD.Sort.NativeOrder	indicates whether sorted parameter values increase or decrease in the root data set for the current product's processing path (either an enterprise L1 or the base *processed* data set from which the current data set is derived, as appropriate)	**code**	C

81.	PD.Sort.NativePri(j)	is the global data ordering priority used when the original tiling was imposed on the precursor to the current data set. If this is a level-1 data set, then PD.Sort.NativePri(j) applies to this data set implicitly.	**number**	C
82.	PD.Sort.OrderFlag(j_tile)	indicates whether the documented sort describes the data in a tile as physically ordered or that of a descriptive vector that contains a list of row index values that if followed would result in retrieving data in a sorted manner.	**code**	C
83.	PD.Sort.OrderVec.*TileNum*(j)	provides an ordered list of tile row numbers corresponding to a documented sort order that differs from the physical storage/stream order.	**number**	C
	PD.TileMap.*	"TileMap" property of PD Items associated with mapping the data across internal tiles.	84-113	
84.	PD.TileMap.AdjacencyPts(n_tile)	gives the list of the horizontal tile locations on a tile grid that have actual point bearing tile data.	**number**	
85.	PD.TileMap.AdjacencyWFs(n_tile)	gives the list of the horizontal tile locations on a tile grid that have actual waveform bearing tile data.	**number**	C
86.	PD.TileMap.ChanList(j_tile,k)	gives the bounds of the occurring array range for each tile.	**number**	C
87.	PD.TileMap.DedictdTile.Cnstrnt1.Name	gives the identifying code of the first parameter that is a single value for every tile when a "dedicated tile" strategy is used.	**code**	C
88.	PD.TileMap.DedictdTile.Cnstrnt1.Val(j)	gives the value of the first constrained data item for the jth tile group.	**number**	C
89.	PD.TileMap.DedictdTile.Cnstrnt2.Name	gives the identifying code of the second parameter that is a single value for every tile when a "dedicated tile" strategy is used.	**code**	C
90.	PD.TileMap.DedictdTile.Cnstrnt2.Val(j)	gives the value of the second constrained data item for the jth tile group.	**number**	C
91.	PD.TileMap.DedictdTile.Cnstrnt3.Name	gives the identifying code of the third parameter that is a single value for every tile when a "dedicated tile" strategy is used.	**code**	C
92.	PD.TileMap.DedictdTile.Cnstrnt3.Val(j)	gives the value of the third constrained data item for the jth tile group.	**number**	C
93.	PD.TileMap.DedictdTile.TileNumValHi(j)	is an integer corresponding to the highest sorting value (i.e., last) of *TileNum* in dedicated tile group(j).	**number**	C
94.	PD.TileMap.DedictdTile.TileNumValLow(j)	is an integer corresponding to the lowest sorting value (i.e. first) of *TileNum* in dedicated tile group(j).	**number**	C
95.	PD.TileMap.GlblPtTyping	encodes the mix of lidar points and synthetic lidar points in the data set.	**code**	E

96.	PD.TileMap.Lat(j_tile,k)	gives the latitude of the northernmost and southernmost data point in the tile	**number**	C (L2)
97.	PD.TileMap.Lon(j_tile,k)	gives the longitude of the westernmost and easternmost data point in the tile	**number**	C (L2)
98.	PD.TileMap.MaxNsamps.*TileNum*	gives the number of samples in the longest waveform vector of tile *TileNum*.	**number**	C
99.	PD.TileMap.N_Rows(j_tile)	is the number of rows of data in tiles indexed by j_tile.	**number**	E
100.	PD.TileMap.N_Tiles	provides the total number of tiles in the data set. This is the top value of j_tile, which ranges over all of the tile identifiers.	**number**	E
101.	PD.TileMap.PtType(j_tile)	indicates whether a point tile's contents are lidar points or synthetic lidar points.	**code**	C
102.	PD.TileMap.PulseRange(j_tile,k)	indicates the temporal boundaries of the data within each tile.	**number**	C
103.	PD.TileMap.RayRange(j_tile,k)	provides bounding values of "iray", the ray index for waveform bearing tiles.	**number**	C
104.	PD.TileMap.Return_Range(j_tile,k)	provides the range of returns in the j_tileth tile.	**number**	C
105.	PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)	supports temporal mapping and reducing the IPP wordsize in tiles populated with IPP values assigned to point or waveform clip returns by tracking the global IPP count from the first IPP (indexed as 0) of the collection data set to the first IPP in each intra-tile time segment of wide band data in tile *TileNum*.	**number**	C
106.	PD.TileMap.TileDelT	provides the number of pulses per tile when a fixed number of pulses per tile tiling scheme is in use.	**number**	C
107.	PD.TileMap.TileNum(j_tile)	provides the mapping between the ordinal index of a tile and the evaluated *TileNum* sub-strings in tags for tiled data items.	**Char string**	E
108.	PD.TileMap.MGRS_Zone(j_tile)	provides the MGRS zone identifier associated with tile j_tile when the Cartesian frame origins are chosen to match MGRS zone location by tile.	**Char string**	C
109.	PD.TileMap.X_Range(j_tile,k)	gives the tile based Cartesian bounds in the X (easting measurement) direction.	**number**	C
110.	PD.TileMap.Y_Range(j_tile,k)	gives the tile based Cartesian bounds in the Y (northing measurement) direction.	**number**	C
111.	PD.TileMap.Z_Range(j_tile,k)	gives the tile based Cartesian bounds in the Z (easting measurement) direction.	**number**	C
112.	PD.TileMap.T_Range(j_tile,k)	gives the temporal range of each tile of data in the data set. Should base T on start time for L1,2,3. Need a L4 convention to handle multiple start time bases. Perhaps should use "absolute" time for L4.	**number**	C
113.	PD.TileMap.WF_Returns(ich, j)	gives the number of tiles at each waveform return level for each array.	**number**	C

	PD.IntTileMap.*	"IntTileMap" property of PD Flags associated with redundancy control in tile population.	114-145	
114.	PD.IntTileMap.RtnsCode.*TileNum*.Flg	indicates the actual number of rows in the tiled data array [either ES.Point.RtnsCode.*TileNum*(ipt,j) or ES.WFClp.RtnsCode.*TileNum*(ip,iray,j)] when repetition factors are used for returns code data in the tile indicated by the value of *TileNum*.	**number**	C
115.	PD.IntTileMap.PulseNum.*TileNum*.Flg	indicates the actual number of rows in the tiled data array when repetition factors are used for pulse number data in the tile indicated by the value of *TileNum*.	**number**	C
116.	PD.IntTileMap.ArrayNum.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, PF.Point.IrayNum.*TileNum*(ipt,j), when repetition factors are used for array number data in the tile indicated by the value of *TileNum*.	**number**	C
117.	PD.IntTileMap.ChNum.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, ES.Point.ChNum.*TileNum*(ipt,j), when repetition factors are used for channel number data in the tile indicated by the value of *TileNum*.	**number**	C
118.	PD.IntTileMap.DirCase.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Array.DirCaseNum.*TileNum*(ip,j), when repetition factors are used for scan phase data in the tile indicated by the value of *TileNum*.	**number**	
119.	PD.IntTileMap.PolCode.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Pulse.PolCode.*TileNum*(ip,j), when repetition factors are used for polarization code data in the tile indicated by the value of *TileNum*.	**number**	C
120.	PD.IntTileMap.FilterCode.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Pulse.FilterCode.*TileNum*(ip,j), when repetition factors are used for filter code data in the tile indicated by the value of *TileNum*.	**number**	C
121.	PD.IntTileMap.RecordStrt.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Pulse.RecordStrt.*TileNum*(ip,j), when repetition factors are used for recording start data in the tile indicated by the value of *TileNum*.	**number**	C
122.	PD.IntTileMap.Scan.Num.*TileNum*.	indicates the actual number of rows in the	**number**	C

	Flg	tiled data array, SC.Pulse.Scan.Num.*TileNum*(ip,j), when repetition factors are used for scan number data in the tile indicated by the value of *TileNum*.		
123.	PD.IntTileMap.Scan.Phase.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Pulse.Scan.Phase.*TileNum*(ip,j), when repetition factors are used for scan phase data in the tile indicated by the value of *TileNum*.	**number**	C
124.	PD.IntTileMap.NadirClass.*ChNum*.Flg	indicates the actual number of rows in the array, SC.Scan.NadirClass.*ChNum*(is) when repeat factors are used.	**number**	C
125.	PD.IntTileMap.NadirClass.*ChNum*.Reps(j)	indicates the number of times the jth unique scan nadir class value in SC.Scan.NadirClass.*ChNum*(is) is repeated in consecutive scans.	**number**	C
126.	PD.IntTileMap.Origin.E_X.Flg	indicates the actual number of rows in the array, ES.Origin.X_ECEF(j_tile) when repeat factors are used.	**number**	C
127.	PD.IntTileMap.Origin.E_Y.Flg	indicates the actual number of rows in the array, ES.Origin.Y_ECEF(j_tile) when repeat factors are used.	**number**	C
128.	PD.IntTileMap.Origin.E_Z.Flg	indicates the actual number of rows in the array, ES.Origin.Z_ECEF(j_tile) when repeat factors are used.	**number**	C
129.	PD.IntTileMap.Origin.E_X.Reps(j)	indicates the number of times the jth unique ECEF referenced origin X coordinate value is repeated in consecutive tiles.	**number**	C
130.	PD.IntTileMap.Origin.E_Y.Reps(j)	indicates the number of times the jth unique ECEF referenced origin Y coordinate value is repeated in consecutive tiles.	**number**	C
131.	PD.IntTileMap.Origin.E_Z.Reps(j)	indicates the number of times the jth unique ECEF referenced origin Z coordinate value is repeated in consecutive tiles.	**number**	C
132.	PD.IntTileMap.Origin.Z_X.Flg	indicates the actual number of rows in the array, ES.Origin.X_MGRS(j_tile) when repeat factors are used.	**number**	C
133.	PD.IntTileMap.Origin.Z_Y.Flg	indicates the actual number of rows in the array, ES.Origin.Y_MGRS(j_tile) when repeat factors are used.	**number**	C
134.	PD.IntTileMap.Origin.Z_Z.Flg	indicates the actual number of rows in the array, ES.Origin.Z_MGRS(j_tile) when repeat factors are used.	**number**	C
135.	PD.IntTileMap.Origin.Z_X.Reps(j)	indicates the number of times the jth unique zone referenced origin X coordinate value is repeated in consecutive tiles.	**number**	C
136.	PD.IntTileMap.Origin.Z_Y.Reps(j)	indicates the number of times the jth unique zone referenced origin Y coordinate value is	**number**	C

		repeated in consecutive tiles.		
137.	PD.IntTileMap.Origin.Z_Z.Reps(j)	indicates the number of times the jth unique zone referenced origin Z coordinate value is repeated in consecutive tiles.	**number**	C
138.	PD.IntTileMap.Pos.*TileNum*.Flg	indicates the actual number of rows in the tiled data array when repetition factors are used for position data in the tile indicated by the value of *TileNum*.	**number**	C
139.	PD.IntTileMap.Intensity.*TileNum*.Flg	indicates the actual number of rows in the tiled data array when repetition factors are used for intensity data in the tile indicated by the value of *TileNum*.	**number**	C
140.	PD.IntTileMap.Nsamps.*TileNum*.Flg	indicates the actual number of rows in the tiled data array when repetition factors are used for waveform data vector length data in the tile indicated by the value of *TileNum*.	**number**	C
141.	PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k), when repetition factors are used for detector array element sensitivity data.	**number**	C
142.	PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(j)	provides the number of repeats for the jth sub-array sequence of unique values of the element-sensitivity item data matrix, SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k), when internal tile mapping is used to reduce redundancy due to values not changing over extended pulse sequences.	**number**	C
143.	PD.IntTileMap.TruthTags.*TileNum*.Flg	PD.IntTileMap.TruthTags.*TileNum*.Flg indicates the actual number of rows in the tiled data array, ES.Point.TruthTags.*TileNum*(ipt,j), when repetition factors are used in populating truth tag data in the tile indicated by the value of *TileNum*.	**number**	C
144.	PD.IntTileMap.Tstart.*TileNum*.Flg	indicates the actual number of rows in the tiled data array when repetition factors are used for waveform clip start time data in the tile indicated by the value of *TileNum*.	**number**	C
145.	PD.IntTileMap.TxCount.*TileNum*.Flg	indicates the actual number of rows in the tiled data array, SC.Pulse.TxCount.*Il.TileNum*(ip,j), when repetition factors are used for IPP duration data in the tile indicated by the value of *TileNum*.	**number**	C
	PD.UserDefined	"User Defined" property of PD Supplies defining metadata for user defined fields	146-151	
146.	PD.UserDefined.NB_Def.P*n*	is the definition text for the user defined narrow band parameter identified by P*n*.	**Char string**	C

147.	PD.UserDefined.NB_Type.P*n*	is a code for the data type (representation) of user defined narrow band data item P*n*.	**code**	C
148.	PD.UserDefined.PtDef.P*n*	is the definition text for the user defined wide band point oriented parameter identified by P*n*.	**Char string**	C
149.	PD.UserDefined.PtType.P*n*	is a code for the data type (representation) of user defined wide band point oriented data item P*n*.	**code**	C
150.	PD.UserDefined.PulseDef.P*n*	is the definition text for the user defined wide band pulse oriented parameter identified by P*n*.	**Char string**	C
151.	PD.UserDefined.PulseType.P*n*	is a code for the data type (representation) of user defined wide band pulse oriented data item P*n*.	**code**	C
	PD.WFClp	"WFClp" property of PD Supplies defining metadata for waveform clip fields	152-153	
152.	PD.WFClp.Nsamps.*TileNum*(ipi,iray,j)	gives the number of samples in each waveform clip in a tile by *TileNum*, which is specific to a return number.	**number**	C
153.	PD.WFClp.PulseNum.*TileNum*(ipi)	assigns an index, ipi, to each entry in the list of "used pulses" with populated waveform clips in a tile by *TileNum*, which is specific to a return number.	**number**	C
	PD.WordSize.*	"Word Size" property of PD Supplies the current data set value for word sizes for variable bit sized data items.	153-170	
154.	PD.Wordsize.ChCombFunc	provides the number of characters in the PF.Channlzn.Hist.Comb.Func(k) strings.	**number**	C
155.	PD.Wordsize.InfAssur	documents the number of bytes in the MD.InfAssur.* bit strings.	**number**	C
156.	PD.Wordsize.Intnsy	documents the number of bytes in the LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) or LD.Point.Intensity.*TileNum*(ipt, j) arrays.		
157.	PD.Wordsize.Iray	documents the number of bytes in the ray identifier for wide band points and waveform clips.	**number**	C
158.	PD.Wordsize.Meta(j)	provides the number of characters in several metadata character stings	**number**	C
159.	PD.WordSize.NB_Def.P*n*	gives number of characters in the user definition supporting user defined narrow band items in the data set.	**number**	C
160.	PD.WordSize.Pos	documents the number of bytes in the X,Y, and Z elements of wide band point coordinates.	**number**	C
161.	PD.WordSize.SensorPos	documents the number of bytes in the stored integer X,Y, and Z elements of the pulse indexed sensor position data.	**number**	C
162.	PD.WordSize.PtDef.P*n*	gives number of characters in the user	**number**	C

		definition supporting user defined wide band point oriented items in the data set.		
163.	PD.WordSize.Pulse	documents the number of bytes in ES.Point.PulseNum.TileNum(ipt, j)	**number**	C
164.	PD.WordSize.PulseDef.P*n*	gives number of characters in the user definition supporting user defined wide band pulse oriented items in the data set.	**number**	C
165.	PD.WordSize.Recon	documents the number of characters in processing algorithm input data lists.	**number**	C
166.	PD.WordSize.RxDatVec	provides the number of bits per sample of data in a received waveform fast time data vector.	**number**	C
167.	PD.WordSize.TileNum	gives number of characters in the evaluated *TileNum* substrings used in the data set.	**number**	E
168.	PD.Wordsize.Truth(j)	gives number of characters in the kmz truth illustration strings.	**number**	C
169.	PD.Wordsize.TxDat	gives the number of bits per sample of SC.WF.TxProfile.Dat(il,j).	**number**	C
170.	PD.Wordsize.Zones	gives number of characters in the MGRS zone designators used in the current data set	**number**	C

9.5. SC –Sensor Configuration

Sensor configuration data describes the primarily level-1 data needed for calibration (in the wide sense) of raw data and the assembly of raw data so that scene mapping can take place. This information is also needed as "just in case" support at L-2. The spatial layout of sensor responses on the pulse, scan, subswath and swath level are laid out in SC.* data. Fast time and slow time sensor dependencies are provided. Some SC.* is inherited to higher enterprise product levels.

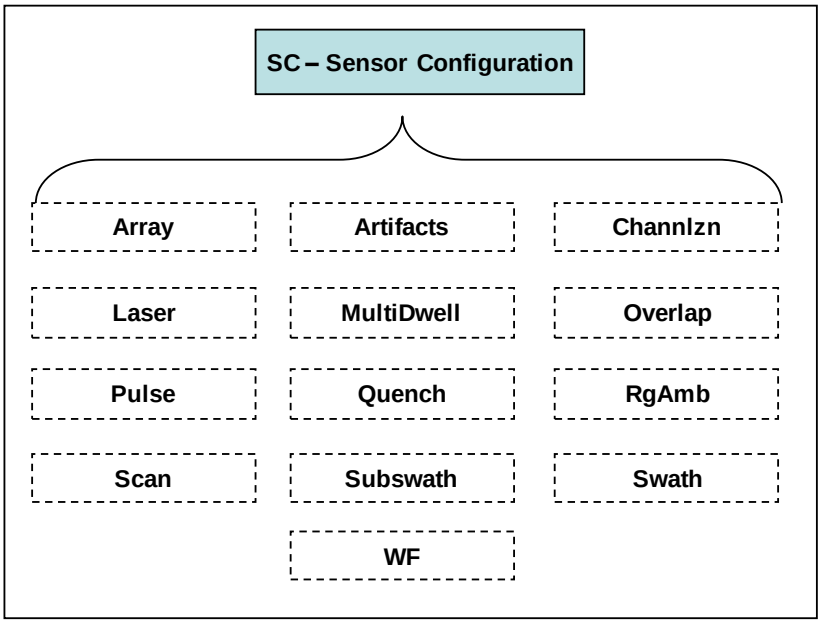


Figure 9.5- 1: SC.* Sensor Configuration second level data structure

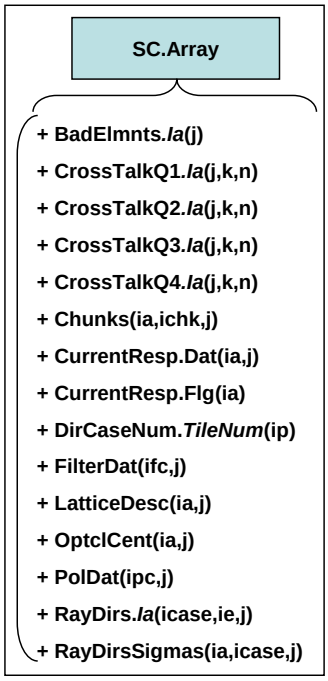


Figure 9.5- 2: Array elements for Sensor Configuration

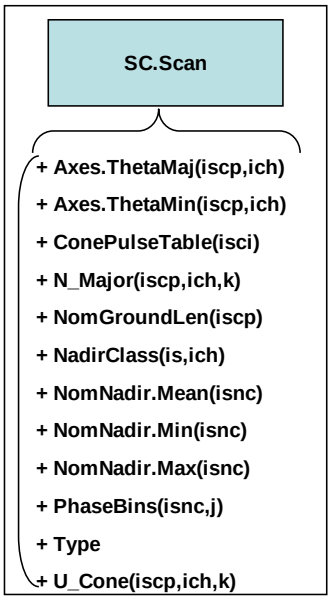


Figure 9.5- 3: Scan elements for Sensor Configuration

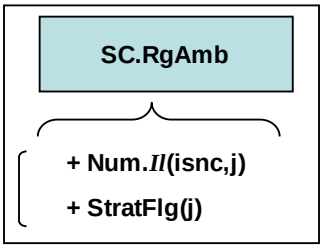


Figure 9.5- 4: RgAmb elements for Sensor Configuration

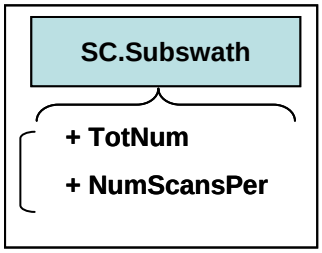


Figure 9.5- 5: Subswath elements for Sensor Configuration

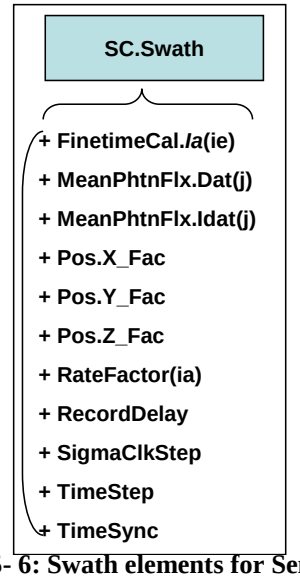


Figure 9.5- 6: Swath elements for Sensor Configuration

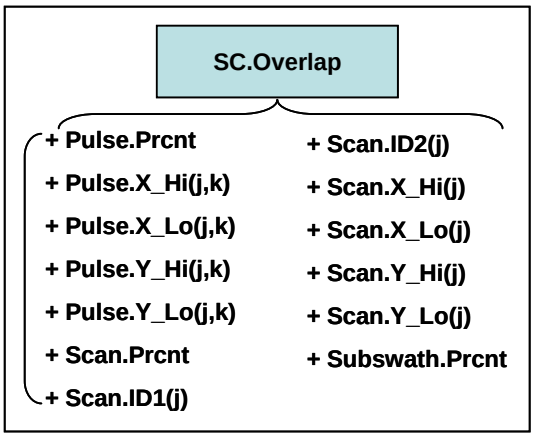


Figure 9.5- 7: Overlap elements for Sensor Configuration

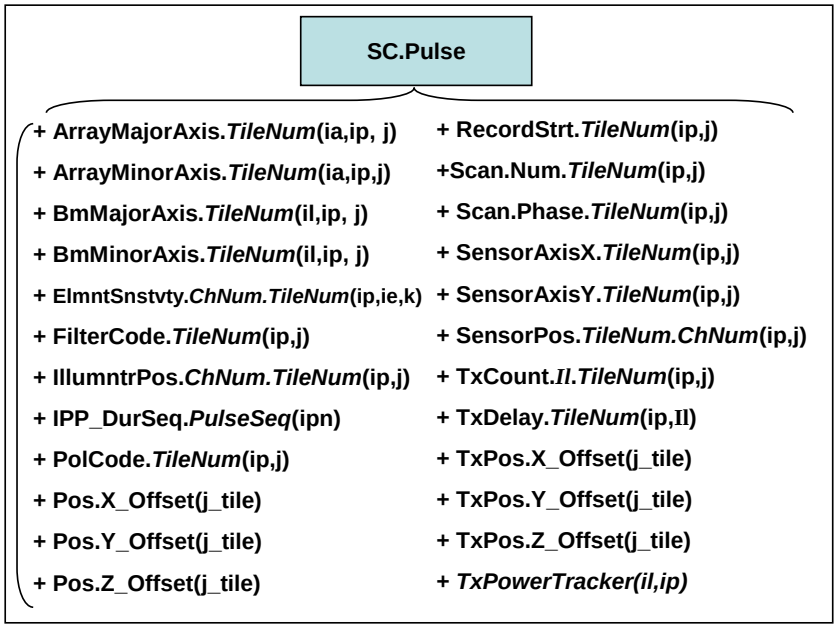


Figure 9.5- 8: Pulse elements for Sensor Configuration

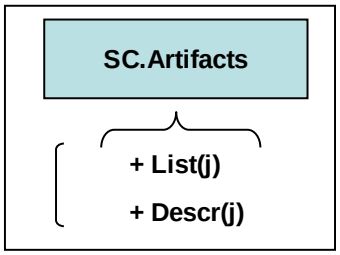


Figure 9.5- 9: Artifact elements for Sensor Configuration

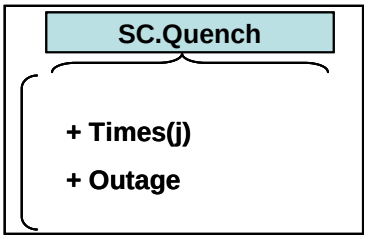


Figure 9.5- 10: Quench elements for Sensor Configuration

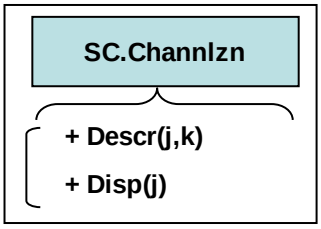


Figure 9.5- 11: Channelzn elements for Sensor Configuration

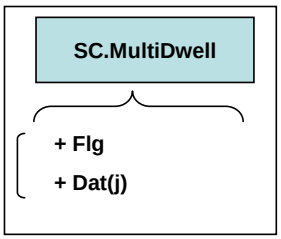


Figure 9.5- 12: MultiDwell elements for Sensor Configuration

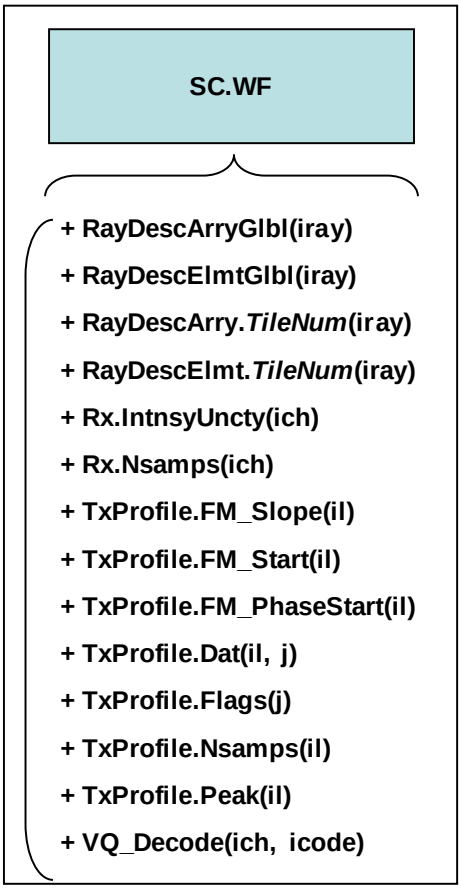


Figure 9.5- 13: WF elements for Sensor Configuration

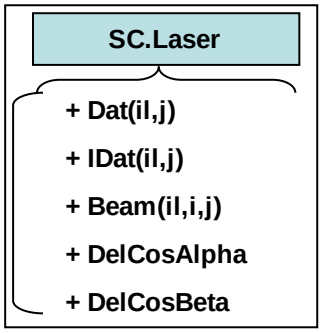


Figure 9.5- 14: Laser elements for Sensor Configuration

Table 6: SC.* --Sensor Configuration

Table Line #	Tag Name	Brief Definition	Domain	Essential/ not
	SC.*	Sensor Configuration class of data	1-105	
	SC.Array.*	"array" property of SC Array level data description items.	1-15	
1.	SC.Array,BadElmnts.*Ia*(j,k)	indicates which elements in each array should be skipped over in processing due to unusable data.	**number**	
2.	SC.Array.CrossTalkQ1.*Ia*(j,k,n)	provides cross talk levels and delays for the lidar's detector array.	**number**	
3.	SC.Array.CrossTalkQ2.*Ia*(j,k,n)	provides cross talk levels and delays for the lidar's detector array.	**number**	
4.	SC.Array.CrossTalkQ3.*Ia*(j,k,n)	provides cross talk levels and delays for the lidar's detector array.	**number**	
5.	SC.Array.CrossTalkQ4.*Ia*(j,k,n)	provides cross talk levels and delays for the lidar's detector array.	**number**	
6.	SC.Array.FilterDat(ifc,j)	specifies the spectral response for the individual optical band pass filters.	**number**	
7.	SC.Array.OptclCent(ia,j)	is the position of the pierce point of the optical axis on the sensing plane.	**number**	
8.	SC.Array.PolDat(ipc, j)	specifies the response for each individual polarizing receive chain filter used in collecting the data for the current dataset.	**number**	
9.	SC.Array.RayDirs.*Ia*(icase,ie,j)	gives the look direction relative to the optical axis as a direction cosine pair for each detecting element in the focal plane array(s), for each optical axis steering case.	**number**	
10.	SC.Array.RayDirsSigmas(ia,icase,,j)	supplies the uncertainty in the look direction for all of the elements in a steering direction case for a given array.	**number**	
11.	SC.Array.LatticeDesc(ia,j)	gives the total number of detecting elements over the lattice chunks and high level lattice chunk description	**number**	
12.	SC.Array.Chunks(ia,ichk,j)	describes the parts of a compound array lattice.	**number**	
13.	SC.Array.CurrentResp.Dat(ia,j)	carries waveform clarity metrics.	**number**	
14.	SC.Array.CurrentRespFlg(ia)	indicates whether the recorded data from the sensor reflects that a sensor photocurrent amplifier has a linear or log (that is, dB) response.	**code**	
15.	SC.Array.DirCaseNum.*TileNum*(ip,j)	provides the value of the steering direction case, which determines which set of array element look directions to use, for each pulse	**number**	
	SC.Artifacts	Artifact metadata describing non-ideal sensor induced data pecularities	16-17	
16.	SC.Artifacts.Descr(j)	describes known issues in the collected data	**Char string**	

17.	SC.Artifacts.List(j)	declares known issues in the collected data	**code**	
	SC.Channlzn	Sensor Channelization metadata (intended for level-1)	18-19	
18.	SC.Channlzn.Descr(j)	associates a lidar measurement functionality with a number associated with a IPP data stream of measurements.	**Char string**	
19.	SC.Channelzn.Disp(j)	documents the intended disposition of the channels in the current data set.	**Char string**	
	SC.Laser.*	"Laser" property of SC. Items associated with specifying the Laser operating/performance parameters	20-24	
20.	SC.Laser.Dat(il,j)	supplies key information about the laser illuminators in the lidar sensor that collected the current data set.	**number**	
21.	SC.Laser.IDat(il,k)	supplies other key information about the laser illuminators in the lidar sensor that collected the current data set as integer codes.	**code**	
22.	SC.Laser.Beam(il,i,j)	is the normalized illuminating beam radiant intensity sampled on a direction cosine grid.	**number**	
23.	SC.Laser.DelCosAlpha	is the beam data grid direction cosine step size for along row increments, which lie along the beam major axis orientation.	**number**	
24.	SC.Laser.DelCosBeta	is the beam data grid direction cosine step size for along column increments, which lie along the beam minor axis orientation.	**number**	
	SC.MultiDwell.*	"MultiDwell" property of SC Items associated with high level scan strategy to revisit scene regions, such as in a geometric diversity collection strategy within a single operation.	25-26	
25.	SC.MultiDwell.Dat(j)	provides summary parameters for multi-dwell data sets at levels below 4.	**number**	
26.	SC.MultiDwell.Flg	is a code indicating the status of the data set with respect to multi-dwell collection strategy.	**code**	
	SC.Overlap.*	"Overlap" property of SC Items associated with overlapping data collection.	27-39	
27.	SC.Overlap.PulsePrcnt	documents collected pulse to pulse instantaneous FOV overlap in support of "frame to frame" registration processing, if it is employed.	**number**	
28.	SC.Overlap.Pulse.X_Hi.(j,k)	gives the nominal along row upper bound of coverage overlap for consecutive pulses.	**number**	
29.	SC.Overlap.Pulse.X_Lo(j,k)	gives the nominal along row lower bound of coverage overlap for consecutive pulses.	**number**	
30.	SC.Overlap.Pulse.Y_Hi(j,k)	gives the nominal cross-row upper bound of coverage overlap for consecutive pulses.	**number**	

31.	SC.Overlap.Pulse.Y_Lo(j,k)	gives the nominal cross-row lower bound of coverage overlap for consecutive pulses.	**number**	
32.	SC.Overlap.ScanPrcnt	documents collected scan to scan overlap to support scan to scan registration, if it is employed.	**number**	
33.	SC.Overlap.Scan.ID1(j)	is a list of scan index values, each of which is an identifier for the first scan in a long list of scan pairs. Taken together with SC.Overlap.Scan.ID2(j), a list of scan pairs is realized. The primary processing operation on these pairs is registration.	**number**	
34.	SC.Overlap.Scan.ID2(j)	is a list of scan index values, each of which is an identifier for the second scan in a long list of scan pairs. Taken together with SC.Overlap.Scan.ID1(j), a list of scan pairs is realized. The primary processing operation on these pairs is registration.	**number**	
35.	SC.Overlap.Scan.X_Hi(j)	is the nominal X- direction upper bound of coverage overlap for a scan pair.	**number**	
36.	SC.Overlap.Scan.X_Lo(j)	is the nominal X- direction lower bound of coverage overlap for a scan pair.	**number**	
37.	SC.Overlap.Scan.Y_Hi(j)	is the nominal Y- direction upper bound of coverage overlap for a scan pair.	**number**	
38.	SC.Overlap.Scan.Y_Lo(j)	is the nominal Y- direction lower bound of coverage overlap for a scan pair.	**number**	
39.	SC.Overlap.SubswathPrcnt	documents collected subswath to subswath overlap in support of subswath registration.	**number**	
	SC.Pulse.*	"Pulse" property of SC Pulse level data items.	40-62	
40.	SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip, j)	gives the ECEF direction of the major axis of the focal plane detector array for each pulse.	**number**	
41.	SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip, j)	gives the ECEF direction of the minor axis of the focal plane detector array for each pulse.	**number**	
42.	SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j)	gives the ECEF direction of the major axis of the transmitted beam for each pulse.	**number**	
43.	SC.Pulse.BmMinorAxis.*TileNum*(il,ip, j)	gives the ECEF direction of the minor axis of the transmitted beam for each pulse.	**number**	
44.	SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)	provides information to make lidar responses uniform over the sensing array and collection period. Consists of corrections to be applied to the received range data vector.	**number**	
45.	SC.Pulse.FilterCode.*TileNum*(ip,j)	carries pulse dependent filter identification in the case that a filter wheel or analogous mechanism is used to provide spectral diversity from pulse to pulse.	**code**	
46.	SC.Pulse.IllumntrPos.*ChNum.TileNum*(ip,j)	gives the channel specific effective optical position of the illuminating laser exit aperture in the ECEF frame.	**number**	

47.	SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn)	provides a sequence of IPP durations in command clock counts that is repeatedly used in a collection operation.	**number**	
48.	SC.Pulse.PolCode.*TileNum*(ip,j)	carries pulse dependent polarizer identification in the case that a filter wheel or analogous mechanism is used to provide polarization from pulse to pulse.	**code**	
49.	SC.Pulse.Pos.X_Offset(j_tile)	carries the x- intercept term in a linear transformation to be applied to the lidar sensor position data stored in a tile to form actual position data in the ECEF frame.	**number**	
50.	SC.Pulse.Pos.Y_Offset(j_tile)	carries the y- intercept term in a linear transformation to be applied to the lidar sensor position data stored in a tile to form actual position data in the ECEF frame.	**number**	
51.	SC.Pulse.Pos.Y_Offset(j_tile)	carries the z- intercept term in a linear transformation to be applied to the lidar sensor position data stored in a tile to form actual position data in the ECEF frame.	**number**	
52.	SC.Pulse.RecordStrt.*TileNum*(ip,j)	is the clock count when the command is issued to commence recording output of the receive electronics.	**number**	
53.	SC.Pulse.Scan.Num.*TileNum*(ip,j)	maps a pulse into a scan number.	**number**	
54.	SC.Pulse.Scan.Phase.*TileNum*(ip,j)	gives the phase of the scan by pulse.	**number**	
55.	SC.Pulse.SensorAxisX.*TileNum*(ip,j)	gives the ECEF components of the sensor frame X axis from which beam steering angles are measured.	**number**	
56.	SC.Pulse.SensorAxisY.*TileNum*(ip,j)	gives the ECEF components of the sensor frame Y axis from which beam steering angles are measured.	**number**	
57.	SC.Pulse.SensorPos.*ChNum.TileNum*(ip,j)	gives the effective optical position of the sensing array.	**number**	
58.	SC.Pulse.TxCount.*Il.TileNum*(ip,j)	is a code supporting the documentation of the duration of all of the IPPs in the collection data underlying the current data set.	**code**	
59.	SC.Pulse.TxDelay.*TileNum*(ip,il)	gives the interval after the command to fire a pulse that the pulse emerges as if from the range of the sensor focal plane.	**number**	
60.	SC.Pulse.TxPos.X_Offset(j_tile)	carries the x- intercept term in a linear transformation to be applied to the lidar illuminator position data stored in a tile to form actual position data in the ECEF frame. Used for bistatic collection only.	**number**	
61.	SC.Pulse.TxPos.Y_Offset(j_tile)	carries the y- intercept term in a linear transformation to be applied to the lidar illuminator position data stored in a tile to form actual position data in the ECEF frame. Used for bistatic collection only.	**number**	

62.	SC.Pulse.TxPos.Z_Offset(j_tile)	carries the z- intercept term in a linear transformation to be applied to the lidar illuminator position data stored in a tile to form actual position data in the ECEF frame. Used for bistatic collection only.	**number**	
	SC.Quench.*		63-64	
63.	SC.Quench.Times(j)	gives the command clock counts when Quenches of Geiger mode APD arrays occur.	**number**	
64.	SC.Quench.Outage	Time in fast data capture clock counts after Quench is initiated when the sensing array becomes reliably sensitive to newly incident photons.	**number**	
	SC.RgAmb.*	"RgAmb" property of SC Items associated with range ambiguous operation of the sensor.	65-66	
65.	SC.RgAmb.Num.*Il*(isnc,j)	is the number of pulses on the propagation path from the sensor to the intended scene range swath at a single instant of time after a steady state has been reached.	**number**	
66.	SC.RgAmb.StratFlg(j)	carries indicators of the range ambiguity management strategy used in the collection.	**code**	
	SC.Scan.*	"Scan" property of SC Scan level data items	67-78	
67.	SC.Scan.Axes.ThetaMaj(iscp,ich)	gives the extent of a scanning maneuver of the optical axis in angle space relative to the sensor.	**number**	
68.	SC.Scan.Axes.ThetaMin(iscp,ich)	gives the extent of a scanning maneuver of the optical axis in angle space relative to the sensor and permits curvature to the scan pattern.	**number**	
69.	SC.Scan.ConePulseTable(isci)	lists the pulse number of the first pulse of each separate scan cone axis parameterization set.	**number**	
70.	SC.Scan.N_Major(iscp,ich,k)	gives the orientation of the scan major axis for each choice of scan cone axis.	**number**	
71.	SC.Scan.NomGroundLen(iscp)	gives the nominal length of a scan as projected to the ground surface from the sensor.	**number**	
72.	SC.Scan.NadirClass.*ChNum*(is)	gives the value of the scan nadir class for each scan.	**number**	
73.	SC.Scan.NomNadir.Max(isnc)	gives the nominal maximum optical axis angle from nadir collected for a scan in each ambiguity number unique scan nadir class.	**number**	
74.	SC.Scan.NomNadir.Mean(isnc)	gives the nominal mean optical axis angle from nadir collected for a scan in each ambiguity number unique scan nadir class.	**number**	
75.	SC.Scan.NomNadir.Min(isnc)	gives the nominal minimum optical axis angle from nadir collected for a scan in each ambiguity number unique scan nadir class.	**number**	

76.	SC.Scan.PhaseBins(isnc,j)	gives the upper bound of scan phase for each ambiguity number specific phase range of a scan for a given scan nadir class.	**number**	
77.	SC.Scan.Type	is a code for the scan trace shape description.	**code**	
78.	SC.Scan.U_Cone(iscp,ich,k)	gives the sensor frame orientation of the jth scan cone parameterization value for the scan cone axis direction.	**number**	
	SC.Subswath.*	"Subswath" property of SC Subswath level data items	79-80	
79.	SC.Subswath.TotNum	gives the number of collection subswaths (native collected scan groups) employed by the collector to create the data set.	**number**	
80.	SC.Subswath.NumScansPer	gives the number of scans per collection subswath.	**number**	
	SC.Swath.*	"Swath" property of SC Swath level data items	81-91	
81.	SC.Swath.FineTimeCal.*Ia*(ie,j)	is a correction that when added to the roundtrip calculation for a return compensates for measured delays within the sensor electronics that bias the roundtrip calculations away from true time.		
82.	SC.Swath.MeanPhtnFlx.Dat(j)	indicates the average photon flux incident on the photo detector array and the control time constant.	**number**	
83.	SC.Swath.MeanPhtnFlx.Idat(j)	describes the SC.Swath.MeanPhtnFlx.Dat	**number**	
84.	SC.Swath.Pos.X_Fac	is the coefficient of the x-axis data in the linear transformations that produce x positions in meters from stored x-components of lidar sensor position.	**number**	
85.	SC.Swath.Pos.Y_Fac	is the coefficient of the y-axis data in the linear transformations that produce y positions in meters from stored y-components of lidar sensor position.	**number**	
86.	SC.Swath.Pos.Z_Fac	is the coefficient of the z-axis data in the linear transformations that produce z positions in meters from stored z-components of lidar sensor position.	**number**	
87.	SC.Swath.RateFactor(ia)	gives a factor that when divided into the command clock step gives the data capture clock step.	**number**	
88.	SC.Swath.RecordDelay	is the number of fast data capture clock cycles after a command is issued to record data within an IPP that recording actually commences.	**number**	
89.	SC.Swath.SigmaClkStep	is the uncertainty in seconds for a sensor command clock step interval.		

90.	SC.Swath.Timestep	is the time in seconds for a sensor command clock step interval.	**number**	
91.	SC.Swath.TimeSync	is the time uncertainty in seconds for start time coordination with UTC or TJD referenced time lines.	**number**	
	SC.WF.*	"WF" property of SC Items associated with waveform of lidar sensor	92-105	
92.	SC.WF.RayDesc.ArryGlbl(iray)	associates an array choice with each a global ray number. Part of a ray definition table that pairs arrays and sensor look directions within the instantaneous field of view.	**number**	
93.	SC.WF.RayDesc.ElmtGlbl(iray)	associates a global array element number with with each ray representing a sensor look direction within the instantaneous field of view.	**number**	C
94.	SC.WF.RayDesc.Arry.*TileNum*(iray)	associates a tile specific ray number with each array choice paired with a particular sensor look direction within the instantaneous field of view.	**number**	
95.	SC.WF.RayDesc.Elmt.*TileNum*(iray)	associates a tile specific array element number with with each ray representing a sensor look direction within the instantaneous field of view that has waveform data included in the tile.	**number**	
96.	SC.WF.Rx.Nsamps(ich)	indicates the length of the receive window in terms of fast time data capture clock intervals.	**number**	
97.	SC.WF.Rx.IntnsyUncty(ich)	Uncertainty in intensity after sensor supplied corrections applied	**number**	
98.	SC.WF.TxProfile.FM_Slope(il)	is the radio frequency slope when linear RF modulation is used to permit lidar pulse compression.	**number**	
99.	SC.WF.TxProfile.FM_Start(il)	is the starting radio frequency when linear RF modulation is used to permit lidar pulse compression.	**number**	
100.	SC.WF.TxProfile.FM_PhaseStart (il)	is the starting phase when linear RF modulation is used to permit lidar pulse compression.	**number**	
101.	SC.WF.TxProfile.Dat(il,j)	SC.WF.TxProfile.Dat(j) is the scaled sampled intensity as a function of (fast) time of the transmitted lidar pulse.	**number**	
102.	SC.WF.TxProfile.Flags(j)	lists a number of codes that describe the transmitted pulse	**code**	
103.	SC.WF.TxProfile.Nsamps(il)	gives the length of the Transmit profile data vector	**number**	
104.	SC.WF.TxProfile.Peak(il)	is the converson factor to remove scaling of SC.WF.TxProfile.Dat(j).	**number**	

105.	SC.WF.VQ_Decode(ia,icode)	Vector quantization look up table for decoding waveforms utilizing pulse compression.	**number**	

9.6. LD –Lidar Data

Lidar data is the wide band data representing processed VRP lidar measurements. Waveform clips (WF_Clp) can be full fast time records of returning pulses or they may be partial subsets of the fast time returns when recorded in linear or pulse compression modes. Points (data items with ".Point." substrings) carry the level-2 and higher result of linear, pulse compression and Geiger Modes when a determination has been made to attribute a particular time or part of a fast time return to a scatterer at a single range. Level-1 data may have short fast time data vectors in the Geiger mode case, but the raw data cannot be attributed to a single range until it is processed to level-2 or higher.

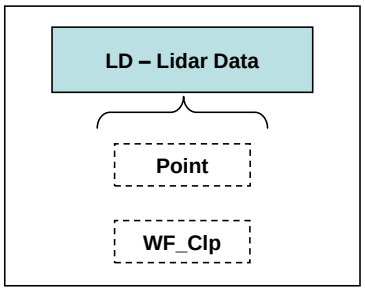


Figure 9.6- 1: LD.* Lidar Data second level data structure

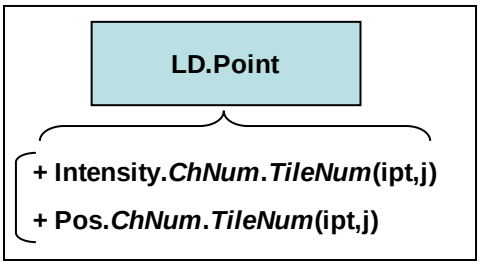


Figure 9.6- 2: Point elements for lidar Data

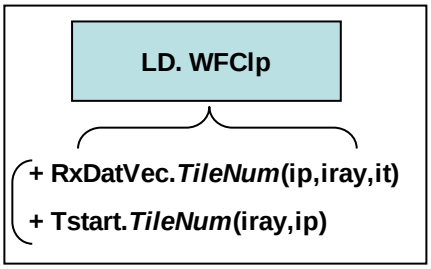


Figure 9.6- 3: WFClp elements for lidar Data

Table 7: LD.* --Lidar Data

Table Line #	Tag Name	Brief Definition	Domain	Essential/ not
	LD.*	Lidar Data class of data	1-4	
	LD.Point.*	"Point" property of LD Point level WB data items	1-2	

1.	LD.Point.Intensity.*ChNum.TileNum* (ipt, j)	gives the intensity value for points in a channel specific data block. This is principally for use at enterprise level-2 where points are dominantly channel unique with a single intensity per channel-point.	**number**	
2.	LD.Point.Pos.*ChNum.TileNum*(ipt,j)	supplies data that can be linearly transformed to earth frame 3-D Cartesian coordinate points in the lidar point cloud within a channel specific tile.	**number**	
3.	LD.SAL.DataGrid.*ChNum.TileNum* (i,j,k)	is a data grid for complex pixel values due to synthetic aperture processing of lidar data.	**number**	
	LD.WFClp.*	"WFClp" property of LD Data items associated with waveform clips of received fast time recorded lidar data.	3-4	
4.	LD.WFClp.RxDatVec.*TileNum*(ip,it ,iray)	carries waveform data	**number**	
5.	LD.WFClp.Tstart.*TileNum*(ip,iray)	gives the fast clock count into the receive window for the beginning of a waveform clip	**number**	

9.7. UD –User Defined Wide Band Data

User defined data may appear at any enterprise level. It is a formal place to add data items not yet specifically defined in this CMMD. The names are not descriptive, but rather generic enumerations of new parameter items that employ the computed substring feature to permit tailoring to user needs. These fields are supported by user provided definition text in PD.UserDefined.*Def.P*n*.

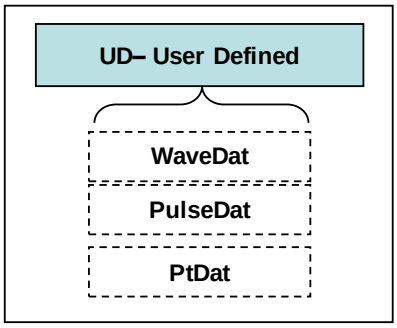


Figure 9.7- 1: UD.* User defined second level data structure

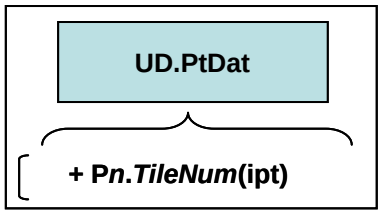


Figure 9.7- 2: UD.* User defined point oriented data structure

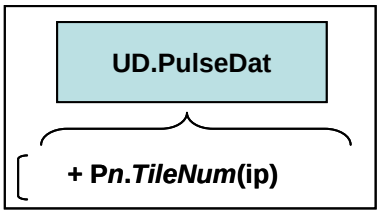


Figure 9.7- 3: UD.* User defined pulse oriented data elelments

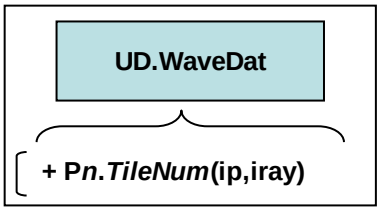


Figure 9.7- 4: UD.* User defined wave oriented data elements

Table 8: UD.* --User Defined Data

Table Line #	Tag Name	Brief Definition	Domain	Essential/ not
	UD.*	lidar Data class of data	1-3	
	UD.PtDat.*	"Point data" property of LD Point level WB user defined data items	1-1	
1	UD.PtDat.P*n.TileNum*(ipt)	carries point oriented wide band user defined data.	**number**	
	UD.PulseDat.*	"WFClp" property of LD Data items associated with waveform clips of received fast time recorded lidar data.	2-2	
2	UD.PulseDat.P*n.TileNum*(ip)	carries pulse oriented wide band user defined data.	**number**	
	UD.WFClpDat.*	"WFClp" property of UD Data items associated with waveform clips of received fast time recorded lidar data.	3-3	
3	UD.WaveDat.P*n.TileNum*(ip,iray)	carries wave clip oriented wide band user defined data. This is a stub for future		

9.8. Creating Profiles of the CMMD

The CMMD, for any given product level, no more specifies a product data set than Webster's dictionary and a grammar book specify a novel like Gone with the Wind. The CMMD establishes a language for lidar data sets that consists of definitions of terms and

how the terms interrelate. Consequently, depending on the "story that is to be told", only a subset of the CMMD terms will necessarily be used in any single data set, even restricting the discussion to enterprise level-2 data sets. Any realized data set is an example of data that follows a *profile* of the CMMD. That profile is a specific subset of data and metadata items selected from the CMMD's dictionary. With a purpose in mind, designers of the profile might use only that subset of the metadata items and data strategies that are permitted by the CMMD that meets their needs.

In any practical lidar data set that complies with the CMMD, some entries in the CMMD might not be present because of one or more of the following reasons:

- The entry was not necessary for the objectives chosen when designing the data set.
- The entry is out of the scope of the objectives, which may be limited due to the known cost of collection, processing or storage and communication constraints on the system that produces a given data set.
- The entry was inconsistent with the entries needed to realize the chosen capability set that has to be supported by the data set.

Consequently, nothing in this CMMD is "mandatory". The CMMD is not a product definition, data set specification or contractually binding document in and of itself. Those items defined within the CMMD that are to be made mandatory for any implementation will be mandated via documents external to the CMMD, which may point to the CMMD for complete conceptual and logical definition. However, some items necessarily for logical closure of the self definition property of CMMD compliant data sets are *essential*. Essential items must be mandated in any implementation spec for lidar data in order that the data set be usable at all.

Beyond the essential data, the population plan for a profile of a product at any given enterprise level, can be determined by a flow down from the capability requirements of the requestor or acquisition authority (whichever applies in the circumstance). The flow down process is made easier by use of the CMMD because an engineer who understands it can use it as a "chinese menu" to order data from. The huge task of enumerating and defining the metadata items from the capabilities that they support has already been performed.

There are basically two approaches to managing profiles. The first is the traditional top down inflexible approach that is the legacy of classical system engineering approaches in the era that preceded open architectures. The second is a lower cost, more adaptable approach which would have been out of the question in the era when interoperability could only result from rigid interfaces that worked in the absence of standards or open publication of the meaning of the individual data items.

In the first case, profiles are defined in a product definition document, often called a "spec" or "requirements document" consisting of a list of items to be populated along with the values of metadata describing the processing that meet the mission objective of the data sets in the product category being profiled. Such profile documents may designate items selected from this CMMD as mandatory. The variety of products that can be profiled is limited by the number of such documents that can be managed by the acquisition authority.

This paradigm is acquisition centric and is prone to the limitations that follow from evolving user needs or institutional barriers to understanding users needs adequately.

In the second case, profiles are self defined on-the-fly by the requestor of the data set. This paradigm is user centric and is able to escape the limitations of the top down approach. Because the CMMD explicitly supports a name-value data population and transfer strategy, it gains phenomenal flexibility and thus can be used to standardize data in either the top down mode or the user centric mode. This is a huge advance over the classical format centric data standarization approaches. Tools can be designed to determine the suitability of data sets based on analysis of the metadata complement and they can be devised to choose a metadata complement based on input capability fields. Configuation management of profile documents is unnecessary in this paradigm. The CMMD does not force the hand of the system designer, rather, it enables a significant choice in profile management.

An overview of the main capabilities and strategies for VRP lidar data sets is shown in figure 9.8-1. This is where the detailed flowdown to a specific profile would begin.

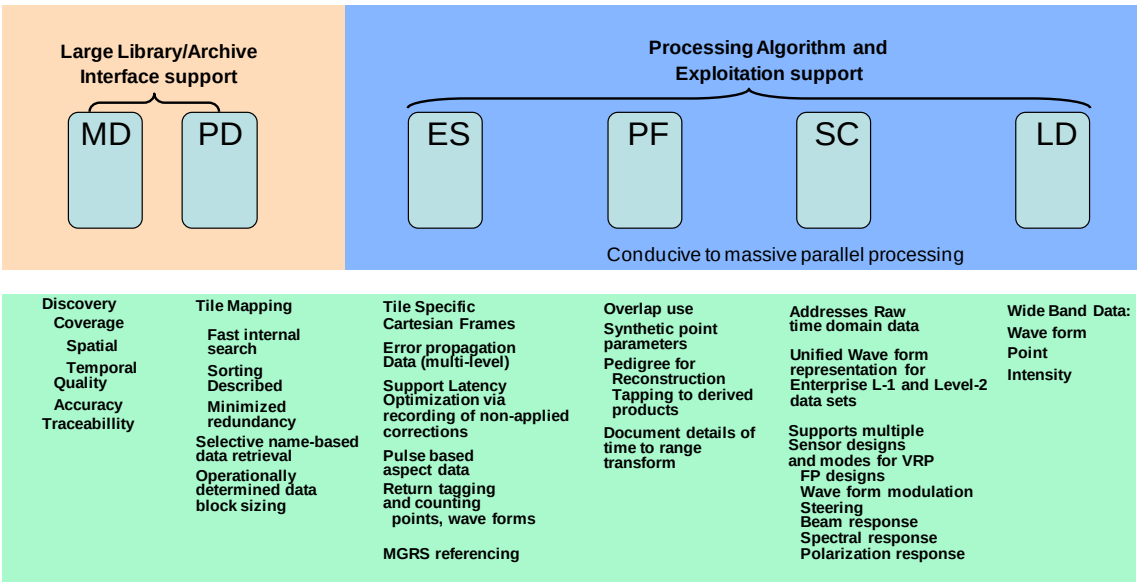


Figure 9.8- 1: Capabilities and strategies derivable from CMMD usage.

In the following figure, the logical steps necessary to create a CMMD lidar data set profile are displayed.

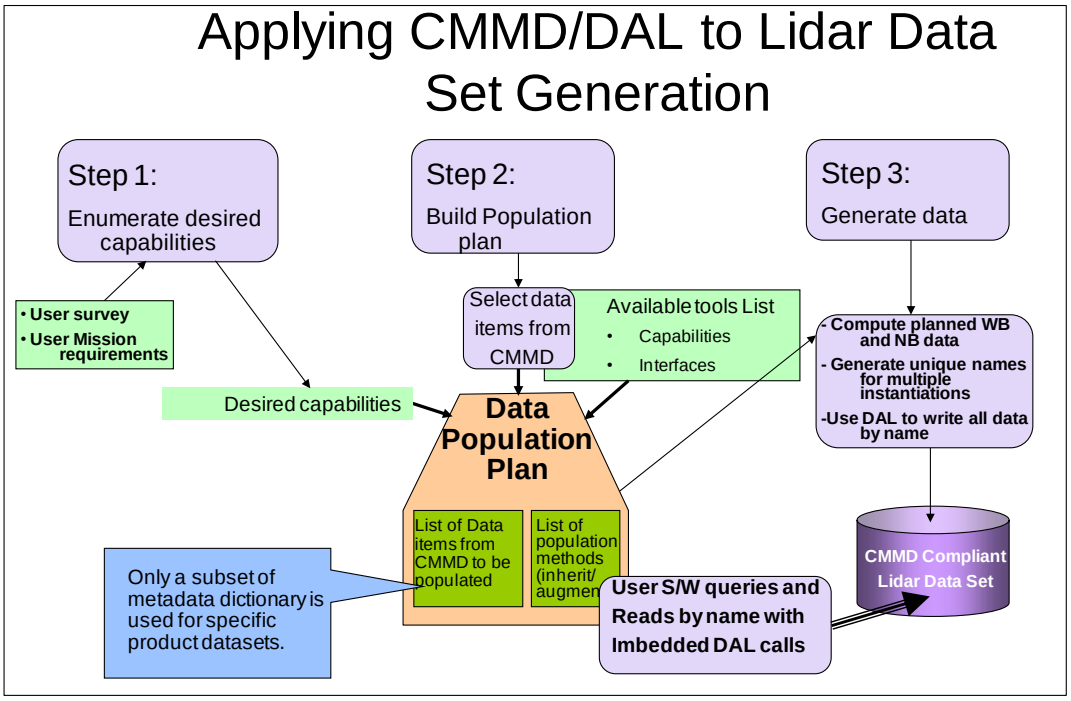


Figure 9.8- 2: Building a Profile based on the CMMD.

An example of the list of metadata that would be populated in a profile of the CMMD that can be part of a spec for a workable CMMD compliant lidar data set is provided in the spreadsheet of Annex B.

10. APPENDIX: Lidar Physical Model Discussion

10.1. Sensor Physics and Physical Operation

10.1.1 Timing Model

10.1.1.1 Time Base, Slow Time, Fast Time

The basic approach is to use 0-based timing from the start of a collection operation and to separately document the start of collection. A Julian date and seconds past midnight time are used for this (see MD.Collect.CollctnDate and MD.Collect.StartTime). The start of collection is the time of command issuance for the first pulse of the first channel to fire . Note that the first channel to "fire" will posess the first pulse repetition interval of a collection operation and that the first pulse of other channels could be delayed slightly compared to the precise start of collection. See SC.Laser.IDat(il,4).

We assume that all lidar designs will use a clock to drive commanded activities such as pulse transmission and the recording of echoes during the anticipated return time interval. Events that can be commanded to happen once [or only a few times] in the interval marked by consecutive pulse transmissions are said to happen in "slow time" and their times of occurrence have intervals adequately measured in **i**nter**p**ulse **p**eriods or IPPs. Platform motion is captured as position and orientation as a function of pulse index, thus being a slow time variable. We use the pulse index, which is incremented at the end of each IPP, to time label events in slow time and the duration of each IPP (denoted τ_{IPP} in this discussion and measured in command clock intervals) is given by SC.Pulse.TxCount.*Il.TileNum*(ip,j). The actual time in seconds for the start of an IPP is the sum of the durations of the preceding IPPs. The length of the clock steps used to mark the times for slow time events is given in seconds by SC.Swath.TimeStep.

On the other hand, to accurately measure the roundtrip times of flight for photons, we must capture times of transmission and arrival much more precisely. Rather than the microsecond scale granularity of "slow time", we capture the time of flight data at the nano-second or better scale of granularity. The events whose times are based on the fine *intrapulse* intervals are said to be measured in "fast time". The fast time increments are a fraction of SC.Swath.TimeStep and are determined by dividing SC.Swath.TimeStep by SC.Swath.RateFactor(ia), where the rate factor values may be specific to each focal plane detector array in the sensor. To keep fast time measurements locked to the command clock, the rate factor will be an integer, so that every slow clock tick occurs at the time of a fast clock tick. [One could design an instrument with a rate factor of unity, but the wordsize for IPP events would become large and present data handling issues.]

Whereas radars are capable of systematically producing a functional transmit time sequence and initiating pulses at a fixed time delay from the computed triggers, the vastly shorter potential lidar pulse durations and the slightly stochastic delay from trigger to pulse emission make it probable that high precision lidars would directly observe their own transmissions to

record a precise emission time in support of the ranging function. This results in a pulse by pulse transmission delay to be recorded.

The pulse dependent transmission delay begins as a level-0 data item. In going to enterprise level-1, all that matters is supporting the reconstruction of the round trip fast time measurement for any pulse at its given slow time index. We can't necessarily leave the transmission delay out of all L-1 and L-2 data sets, however. The reason is that we must be careful about the transmission delay in the case of Nambig>0[16]. The roundtrip time calculation depends on the ambiguity number. The ambiguity number, in turn depends on the assumed desired range over which the mapping is intended to focus. If system accuracy requirements drive incorporation of the effect of varying transmission delays during a single collection, population of SC.Pulse.TxDelay.*TileNum*(ip) supplements the values of SC.Pulse.TxCount.*Il.TileNum*(ip,j) to permit very precise range timing, even for nominally constant PRFs.

It is an easy matter to record transmit times on a pulse by pulse basis regardless of whether the hardware is actually measuring the transmit pulse time directly. Thus, only pulse indexed[17] transmit command timing data are used in the lidar conceptual model and transmit sequence polynomial functions are not used. If a lidar operationally uses such functions in its commanding of transmissions, the commanded clock counts for transmission are evaluated and the values of the interpulse intervals are saved to the lidar data set on a pulse by pulse basis for use in L-1 and L-2 CMMD compliant data sets.

Refer to figure 10.1.1-1, the Interpulse sequence diagram to see the start time and ambiguity number illustrated on the first N **i**nter**p**ulse **p**eriods (IPP) of a lidar collection operation. The operation's internal timing begins with IPP[18]zero and the time since the start of collection operation, t, is set to t=0 at the beginning of IPP zero. This time, t, is incremented in sensor (command) clock counts and the data index, ip (i.e., the pulse index), is the parameter in the data model that tracks the internal sensor command timing. The (L-2) external time corresponding to t=0[19] is given by MD.Aggregate.Parent.StartTime(j)[20] on the date given by MD.Aggregate.Parent.CollctnDate(j). There is a faster clock that is used for fast time data capture. It runs faster by a factor of SC.Swath.RateFactor. The conversion to time in seconds is

[16] If the transmit delays vary from pulse to pulse, then the ambiguity number is needed to determine which transmit delay to use in the round trip time equation for a particular return.

[17] Data stored on a pulse basis is called "pulse indexed", indicating the association of the IPP number with the recorded event such as a pulse transmission command or measurement of sensor position.

[18] One may think of an IPP has having one transmitted pulse (per channel).

[19] This marks the start of the first IPP of the first channel to "fire" in a collection operation. The first IPP of later starting channels, if they exist, begins at clock counts given in SC.Pulse.TxDelay.*TileNum*(ip,il).

[20] Enterprise Level-2 and higher data sets may be temporal subsets whose start time is given in MD.Temporal.Coverage.StartTime, a time later than t=0 for the original collection operation.

given by the parameter SC.Swath.TimeStep, the interval associated with sensor command timing clock.

The IPPs are all the same in structure. That is, they are kicked off at a time of issuance of the command to transmit a pulse. The value of τ_{IPP}, the number of command clock counts between successive transmit commands, is given by SC.Pulse.TxCount.*Il.TileNum*(ip,j)[21]. A short time later, the pulse is emitted and a time is marked when it passes as if emitted at the effective illuminator position. For the common monostatic lidar design case, that will be the position of the detection focal plane. For the bistatic case, the illuminator position is given by SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j). This small transmit time delay interval is given by SC.Pulse.TxDelay.*TileNum*(ip). That interval is recorded as a number of fast data capture clock cycles after the command to transmit. A flash is seen as if emitted from the focal plane position SC.Pulse.TxDelay.*TileNum*(ip) fast data capture clock cycles after the command to transmit. After a ("gate") delay, the receiving window is ordered opened at the record start time, given by SC.Pulse.RecordStrt.*TileNum*(ip,j) in units of command clock intervals. The recording begins after SC.Swath.RecordDelay cycles of the fast data capture clock[22]. Recording continues until SC.Pulse.NSamps clock cycles have passed.

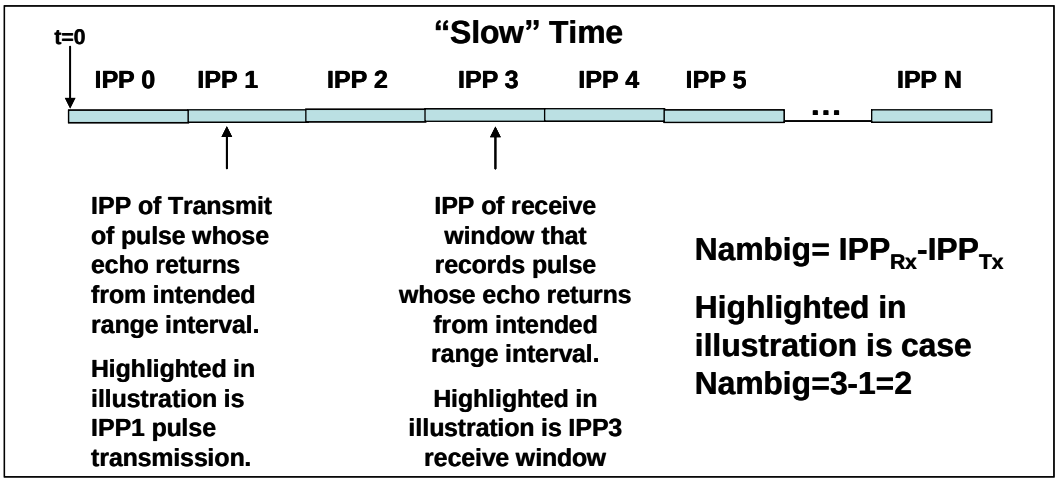


Figure 10.1.1- 1 Slow Time InterPulse Period Sequence Illustration

[21] Note that if we operate at a constant ambiguity number but vary the range significantly, as might happen in scanning the beam, the PRF must vary. Since the time of transmit is given on an IPP basis, we can support PRF variation with the data of this model.

[22] The apparent reversal of time tracking between Tx and Rx is due to the possibility of needing to slide the listen window (or "range gate") within the IPP on a pulse to pulse basis to accommodate scan geometry variation. Unlike the Tx case, the delay in response to the command to record data is a property of the instrument configuration and should be constant over a collection operation.

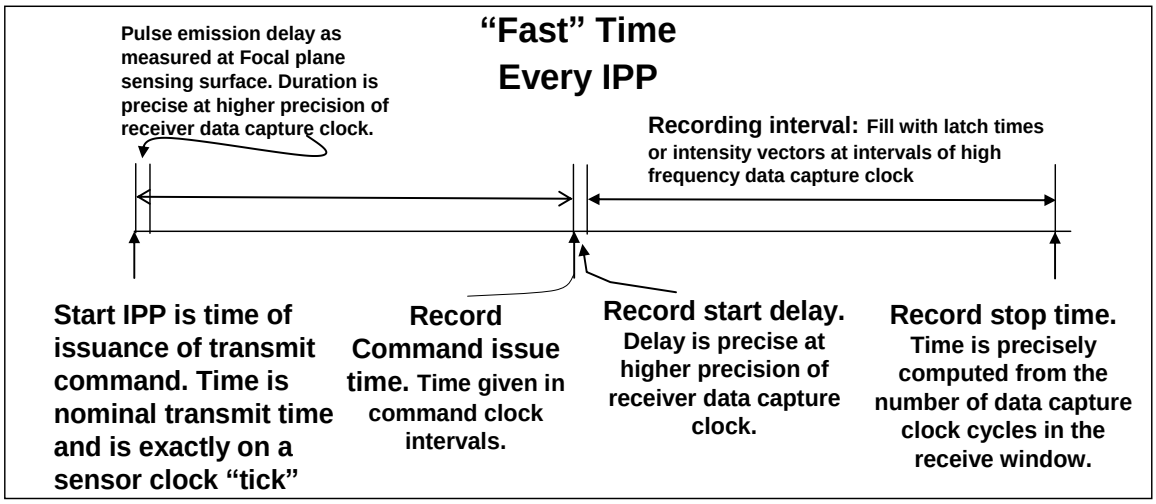


Figure 10.1.1- 2 Fast Time InterPulse Period Structure Illustration

For a graphical explanation of how roundtrip timing is computed from a fast time photon return event(s) recorded in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), see the round trip timing figure in the detailed definition of SC.Pulse.TxDelay.*TileNum*(ip). That figure places more detail within a pulse than summarized in Figure10.1.1-2 and gives the context of Figure 10.1.1-1 as well. The round trip timing diagram illustrates that:

SC.Pulse.TxCount.*Il*.*TileNum*(ip,j) is $\tau_{IPP}(ip) = t_{1ip+1} - t_{1ip}$,
SC.Pulse.TxDelay.*TileNum*(ip)=t_{2ip}-t_{1ip},
SC.Pulse.RecordStrt.*TileNum*(ip)=t_{3ip}-t_{1ip}
SC.Swath.RecordDelay=t_{4ip}-t_{3ip}
LD.WFClp.Tstart.*TileNum*(ip,iray,j)= $t_{4.5ip}$-t_{1ip}, where $t_{4.5}$ signifies a time at or after the start of the record window but before the end of the record window when a sub-interval of the recorded wave form that is in a clip begins.

To prevent the lidar data model of this CMMD from limiting strategies in sensor and mode design that are meant to optimize collection rate, it is necessary that we be able to support variation of the ambiguity number during collection. Recovery of the correct ambiguity number to use for a particular return is described in the discussions of SC.Scan.NadirClass.*ChNum*(is) and SC.RgAmb.Num.*Il*(isnc,j) where we define the nadir classes and store the range ambiguity sequence for the nadir classes respectively. Since there may be 20 to 100 million pulses in a collection operation, we avoid listing the ambiguity number by pulse number. Instead we recover it a very general way from a sequence of table look ups. When there is no ambiguity number

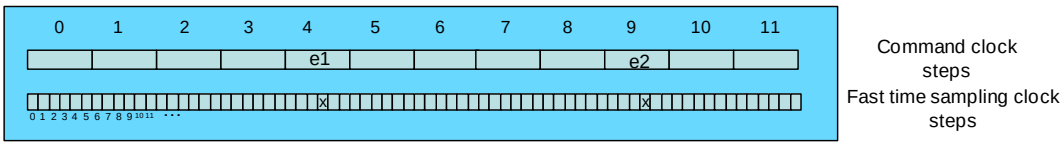
variation within a collection operation, the table scheme collapses to a simple procedure. The following table summarizes the process in the general case:

Determination of the Ambiguity number to be used in Roundtrip Time Calculation given an IPP in which a return is received

Step	Metadata source
0) Choose the IPP number in which a return is received	From processor, not metadata
1)Determine the scan number for an IPP.	SC.Pulse.Scan.Num.*TileNum*(ip, j)
2)Determine the nadir class for the scan	SC.Scan.NadirClass.*ChNum*(is)
3)Determine the scan phase for the IPP.	SC.Pulse.Scan.Phase.*TileNum*(ip, j)
4)Determine ambiguity number phase bin (value of j in step 5) from the max phase for the jth ambiguity number bin of a nadir class.	SC.Scan.PhaseBins(isnc,j)
5) look up IPP specific ambiguity number for a particular illuminator, *Il*.	SC.RgAmb.Num.*Il*(isnc,j)

<u>Note regarding precise time interval calculation in discrete time keeping</u>

Classical physics treats instants in time as if an instant were a point on a line. This is a continuum world model where time can be defined as limited to an interval of zero duration. We speak of time differences as $\tau=t_2-t_1$ and with the continuum and point model, this is unambiguous. However, the world of scientific instruments is a discrete world where time intervals are not zero, but rather come in clock steps. A real system that does its own time keeping can not measure a time interval to finer scale than its clock step interval.



Times are the indices of steps on a time line. The interval between two events, each occurring in a clock time step is $\tau=t_2-t_1$. Illustration: event 1 occurs in interval 4 (at t_1=4). Event 2 occurs in interval 9 (at t_2=9).

Convention:The time between event 1 and event 2 is (9-4)=5. This means that event 2 happens *in* the interval that is the τth step after event 1. The 0th step after e1 is the interval e1 is in. The step immediately after the step e1 is in is the first step . Thus, we define time intervals as if events are centered in clock steps and there is one half of an interval following an event within the same step and one half of an interval preceding an event within the same step. An interval between events is thus one more than the number of integer intervals between the step containing the first event and the step containing the second event. In fact, the clock can not locate an event to finer time granularity than the step that it occurs in. We must identify whether the pertinent steps are those of the command clock or the fast time sampling clock when defining an interval.

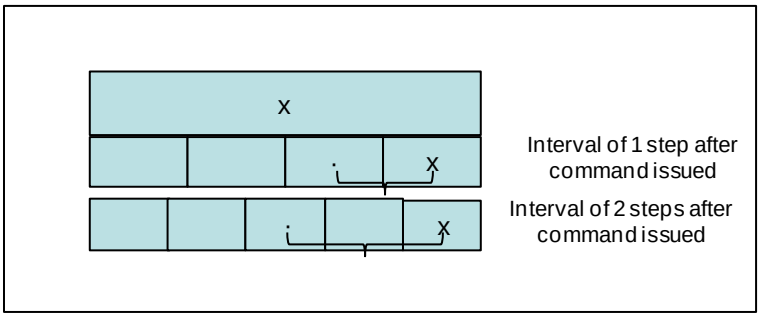
Figure 10.1.1- 3 Discrete time interval convention for determining the time step that is the end of an interval given its start and its duration.

The discussion in Figure 10.1.1-3 leaves open the problem of a command being issued on the command clock time line and the delay being given as a number of fast time sampling steps. This

is the case of SC.Swath.RecordDelay. The convention on counting depends on whether the rate factor, SC.Swath.RateFactor, is odd or even.

The simpler case is when an odd integer number of fast sampling clock steps cover a command clock step. Then the center of a fast time sampling step can occur at the same "continuum" time as the center of the command clock step. The command is interpreted as issued at that fast sampling clock interval. The illustration of figure 10.1.1-3 applies directly. Note that the delays may be negative(!). This is not a break down of causality, but rather the possible result of laboratory calibrations that may determine that command issuance is not truly at the center of the command clock interval.

In the even case, there are an integer number of fast sample clock intervals within the command clock interval following the mid-time of the command clock interval, which is the "continuum" time attributed to the issuance of the command. The number of sampling time steps after the command time begins counting in the fast sample clock interval immediately following the mid time of the command clock interval. It ends counting in the fast time sampling interval of the delayed response to the issued command. Effectively, we treat this case as if the command is issued at the mid time of the fast time clock step immediately *following* the center time of the command clock interval in which the command is issued. We thus count the fast intervals after the command clock interval mid time up to the fast time clock interval of the response of the command and subtract 1.



In either case, conceivably, the delayed response could come in a command clock interval that follows the clock interval of the issuance of the command.

To summarize in words, the slow time model attaches times to pulses, and those times can be used to synchronize the lidar point cloud and wave form range vectors to external activities. The sensor position and attitude (orientation of the line of sight) data are available as direct functions of the pulse number and translation to time is not necessary. To determine the sensor position for collection of a point, the following process is supported in the metadata definitions:

- Points have pulse numbers supplied in ES.Point.PulseNum.*TileNum*(ipt, j). These are start of collection referenced IPP values if PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) is unpopulated. When PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) is given, the values are 0 based at the starting pulse for each point tile. {For the same value of *TileNum*, ES.Point.PulseNum.*TileNum*(ipt, j) has the pulse numbers for the points in LD.Point.Pos.*TileNum*(ipt,j) with the same value of "ipt".}
- Wave clip pulse numbers are defined in PD.WFClp.PulseNum.*TileNum*(ipi)
- Full waveforms are stored by pulse (i.e., IPP number), ip, in each tile of LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)
- Tile level ip can be transformed to global IPP number via IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndex.Tsegts, 3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent, PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases)
- Sensor position at the time of a specific IPP is given by SC.Pulse.SensorPos.*TileNum*(ip,j). Sensor position is given in pulse oriented tiles. Each Sensor position tile runs from a low pulse number to a high pulse number, both relative to the start of the collection. So—
    A. Find the 'start of collection' based pulse number for a point of interest
    B. Look in the tilemap, PD.TileMap.PulseRange(j_tile,k), to see which tile number has that pulse. This requires use of the *TileNum* to j_tile translation in PD.TileMap.TileNum(j_tile).
    C. Select SC.Pulse.SensorPos.*TileNum*(ip,j) value for the needed pulse's (ip) sensor position.

At this point, we need to note that the timing data needs vary with the product level. The level-0 data has the direct instrument outputs and is considered a source for level-1 and higher products. As indicated in the definitions of section 5, we only define data at level-1 and above via the CMMD lidar document series (enterprise L1,L2,L3,L4,L5). At level-2, the ranging will have already been performed for XYZ points, but not for waveform[23] clips. We have to address expected needs for timing data at level-2 on two separate time scales. The first is "slow time", which is the timing granularity needed to synch up the earth frame XYZ data with the collection

[23] But we can store a nominal position value for a clip to help locate the return region within the scene. See ES.WFClp.Locs.*TileNum*(ipi,iray,j)

geometry. The second is fast[24] time, timing data that can be used to recover level-1-like parameterization of the returns or to map L-2 XYZ points back to L-1 data.

On the first time scale, the pulse number can be converted to a pulse time for checking coincidence with other sensors or known scene activity. The primary purpose of slow time data at level-2 is to support reconstruction of collection geometry, and to establish temporal sequence. In particular, when single XYZ position returns are obtained from a diversity of aspect angle, understanding the potential for variation in response due to possible obstructions at only some aspect angles can explain scene properties or act to confirm hypotheses suggested by sparse returns in neighboring regions. So we need slow time parameters pulse number and pulse time to be defined at level-2

The issues for level-2 fast time data are less simple. First of all, the significance of fast time data is different for Geiger mode, linear mode and pulse compression[25] mode.

- For Geiger mode, the time of transmit, time of latch[26], and ambiguity number fix the roundtrip time measurement. The round trip time, in turn, fixes the optical range ("optical" range includes the effects of ray curvature and propagation speed due to refractive index). If we have a level-2 XYZ point or even an intermediate angle, angle, range point, the most we could do with fast time metadata (and a time to range function) is to re-compute the level-1 data from which the point came (and that is just an element identifier and a latching time). If that is the goal, a simpler approach might be to provide metadata that permit associating a level-2 point with its level-1 source element and pulse. Then the inversion calculations do not need to be done --and they could be very nasty if ray curvature is involved. It is doubtful this process would ever be exercised in exploitation because there is no scene related meaning to the latching time other than the implied position of measurement itself, which we had before inversion. Ordinary[27] Geiger mode data is not very "rich" relatively speaking.
- For Linear mode, point extraction from wave forms depends on identifying a time of reception for a point return in the wave form. Point returns are associated with fast time indexes of the wave form sampled at key waveform structural signatures, especially the first time the photocurrent (and possibly simultaneously, the time derivative of the photocurrent)

[24] The terms, "slow time" and "fast time" are borrowed from the synthetic aperture radar community. Long before the introduction of lidars, the radar community found the benefit of these terms to distinguish the pulse scale and sub-pulse scale time dependencies. The same technical issues arise in lidar so these terms are used in the timing model as defined here.

[25] Pulse compression modes are a subset of the linear mode family. They use extended duration pulses and a means to range compress the data to achieve fine range resolution. Pulse compression for lidar has been accomplished via polarization modulation and RF modulation of either the laser line or amplitude of the laser carrier signals.

[26] The moment that the nonlinear, saturated APD/ detector output triggers the recording of a photon arrival event.

[27] More sophisticated uses of gm detection have been proposed that even permit coherent detection. These could thus be richer in information content.

exceeds a threshold within the waveform record. The time marker is used to compute a roundtrip time and a range is thus determined once the refractive effects of propagation have been appropriately applied to correct for the speed of light and ray bending. Given the resulting point, to find the level-1 intensity history that went with that point, we would need the pulse number and the array element identifier. If all registrations (adjustments of the XYZ positions away from the implied offset from the sensor due purely to propagation path and time) can be backed out, the array element that recorded the intensity history that was used to create a point can be determined from the earth frame point position and the earth frame sensor position and attitude, the last two of which can be found via the pulse number. The intensity history is unique to the array element and pulse number. The only way this inversion really differs from the Geiger Mode case is that the linear mode intensity data at level-1 has context and some rule that chose a time out of many possible times to claim for the point. Note that the pulse time determines the sensor position and attitude so that given the refraction model data, we can reconstruct the round trip time interval for a point without using any fast time data. But the roundtrip interval is not sufficient fast time documentation to determine where in the recorded intensity data the roundtrip terminates. To get that information so that the appropriate interval of the recorded intensity could be identified, we need the ambiguity number, transmit times of the source pulse and intervening pulses, and the "gate" delay to the start of the recording window. There is thus justification for carrying this data, whose primary purpose is to make level-1 data processable to points and wave clips, forward to level-2. The richness of the received waveform information in the temporal vicinity of the XYZ point can lead to some assessment of the scene that can't be afforded by Geiger mode data. Thus, there is a conceivable scenario in which the presence of the fast time model parameters at level-2 could be of some value in the context of linear mode. Note that this scenario is based on being able to identify the raw intensity data associated with a point in a point cloud and using level-1 data as the source of that data. The level-2 data, with its fast time model parameters is just a cueing product. The "inversion" can not recreate intensity history data from data at level-2. It just provides a way to locate it. If we had started with the level-1 data alone, we would not know where in space a particular waveform clip was "sounding". So the level-2 data with its fast time metadata together with level-1 data make a useable package[28]. Most likely, the tools to analyze waveform clips would need to be fairly mature before this scenario would ever be exercised in practice. The minimum tool would be a graphic display of the recorded fast-time waveform. This would only be of interest to algorithm developers. For realistic production user exploitation, more advanced interpretive tools will need to be developed.

- For Pulse compression mode, rather than a short pulse of duration the order of the light travel time across a range resolution cell, a very long pulse is encoded in a way that its region of

[28] For particular points of interest, the processing can retain waveform clips in L-2 data sets and avoid recourse to L-1 data for those cases.

scattering can be assigned to a spatial interval that is very small relative to the distance given by the product of the pulse duration and the speed of light. Consequently, an XYZ point is something that might be assigned to a position within a range resolution cell, but the data and signal processing can only localize the return to a range resolution cell, not a specific value of X,Y, and Z. This is less different than points determined from Geiger mode data than it seems at first blush. The Geiger mode data only appears to be defined at a point. In fact, there is an uncertainty associated with each point measuring position for Geiger mode too. It just needs to be imposed after a nominal range has been assigned, while in the case of pulse compression, the width of the resolution cell in range is part and parcel of the pulse compression process. If we had XYZ point data and wish to "disperse" it back to the time domain of the transmitted pulse, we would not only need roundtrip timing parameters but additional waveform parameters to determine the support region for a return in the dispersed domain. After dispersing a particular return, it can be reweighted and compressed to obtain different related spatial domain representations of the scene. [Commonly, side lobe suppression vs main lobe width could be a desired trade.] Thus, for pulse compression mode lidar, there is also a use scenario for fast time parameters at level-2, but unlike the direct time of flight timing linear mode case, we need not have the level-1 data for recourse. We can reconstruct the dispersed domain data from the support data and adequately sampled level-2 product. This means that more than just the XYZ point cloud would need to be available in the level-2 product. A gridded product with sample spacing denser than one per resolution cell would be needed and usable[29].

[29] For the version 1.0 enterprise L-2 model, we define a grid only in the range direction of a sensor based angle-angle-range system. If it proves to be a worthwhile extension, a version 1.x could add an XYZ earth frame grid in the future. This decision is based on the key utility of reversion of range data being served by the range axis gridding and currently perceived low prospects for exploitation value of lidar response over a fine 3-D grid.

MD.* Discovery Time Metadata

The above discussion focused on the detailed timing data needed to process raw data and to assure that the line of sight could be reconstructed for any point in an enterprise L-2, L-3, or L-4 point cloud. There is, in addition, a general discovery/data selection time description problem that requires that we trace the temporal coverage through the data chain from L-1 to L-5. The observation is that original collection data is put into enterprise L-1 data sets. Level-2, 3, 4 data sets are all *processed* [not collected] from lower level data and may have more restricted temporal coverage than the parent data set(s) from which each is created. All L-2 and L-3 data sets processed from a given L-1 data set have the same collected data basis as the L-1 and thus share the collection start date, start time, [as well as operation identifier, platform identifier, sensor ID and sensor type]. So we specify these items as original values in [MD.Collect.CollctnDate, MD.Collect.CollctnOpID, MD.Collect.StartTime, MD.Collect.PlatformID, MD.Sensor.lidarType] and as inherited items in [MD.Aggregate.Parent.CollctnDate(j), MD.Aggregate.Parent.CollctnOpID(j), MD.Aggregate.Parent.PlatformID(j), MD.Aggregate.Parent.SensorID(j), and MD.Aggregate.Parent.SensorType(j)].

In addition, when creating a data set from one or more parent data sets, it may be useful to select a sub-interval of collection time to populate the output data set at any enterprise level from 2 to 5. Thus, the time coverage in a data set is not fully described by the time data for its parent(s). To complete the description, we add MD.Coverage.Temporal.Duration, MD.Coverage.Temporal.StartDate, and MD.Coverage.Temporal.StartTime, which apply to the current data set rather than the parents or the L-1 original collected data.

10.1.1.2 Temporal Strategy

This CMMD provides the metadata “tools” to support a range of options on how the data may be organized for direct viewing by a display tool. Of special interest is the temporal organization of the data.

Data may be collected in modes with 3 general characterizations of native temporal clustering.

1. Data in a contiguous spatial region are collected over a single very short time interval (as in area mapping strategies)
2. Data in a contiguous spatial region are collected over a single rather extended time interval (as in monitoring or multi-aspect strategies)
3. Data in a contiguous spatial region are collected over multiple time intervals with gaps in temporal coverage within an extended period. This is as if we look at a place, look at another place and then return to the first place.

On the other hand, there are two basic viewing approaches to the temporal details of the data:

1. Ignore the temporal sequence and view the integrated scene as a single whole.

2. Slice the data up into temporal frames and display those frames in a sequence (video option)

Of course, non-temporal options exist which may or may not map directly into one of the above by reason of the native collection strategy. For example, a scene may be viewed from a particular aspect in a display device, and the user may wish to display only those points actually collected from the azimuth viewed in an angular "frame". We are not addressing that in this section.

This CMMD supports both the labeling of point data by the pulse on which it was collected and the sorting of the data on the basis of those values. Furthermore, tiling can be done so that each single tile has points only collected within a specific time interval and the tile mapping metadata supports documenting the tiles by time of collection sequencing. This creates the possibility of a video like display of 3-D point clouds. Depending how the data was collected, the field of view in such a temporal sequence of views will either rove about the scene or stare at a particular volume.

PF.Sort.PointPriority.*TileNum*(j) provides the priorities by which point data are sorted within a given tile [e.g. first by time, then by return]. This would be of key use when tiles and time slice frames are not aligned at the frequency at which a user wishes to view the temporal dependence of the scene of interest. On the other hand, if the processing has been done to support viewing of a 3-D point cloud as a video at the integration time per frame that a user wishes, the key metadata will be PD.TileMap.PulseRange(j_tile,k), which indicates the temporal boundaries of the data within each tile.

Insight into how the data was ordered prior to the first time it was tiled in creating a CMMD compliant product chain is given by PD.Sort.NativePri(j). This gives the global data ordering priority existing when the original tiling was imposed on the original precursor to the current data set. It could be the case that the processing did not change the native ordering before tiling, so this is a possible source of information about the collection ordering of the data. Definitive data about the collection operation is to be found in, for example, SC.MultiDwell.Dat(j), which provides summary parameters for multi-dwell data sets at levels below 4, and in SC.Scan.Type, which carries a code for the scan trace shape description.

10.1.1.3 Structural Unification of Linear and Geiger Mode Data

While the structural unification of linear and Geiger mode in the data model is significant for enterprise level-1 data, this elegance has no impact at level-2 and above. At level-2, the use of LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) is exclusively applicable to extended waveform clips, so the dual capability is of no impact.

10.1.2 Beam Model

Normal laser sources naturally have a narrow beam, both in the size of the resonating cavity exit aperture and in the divergence angle of that emerging beam. Consequently, in order to cover significant amounts of ground some method of spreading and/or scanning the illuminating laser beam may be necessary (as in short range applications when the sensor is using an array to "image" the illuminated beam footprint). The measured position of a scatterer is determined from the fundamental measurement of a time[30] and a direction. Where within a receive beam[31] the energy of a return arrives and the orientation of the receive beam axis combine to specify an angular coordinate to the "ray" from the sensor to measurement positions. The round trip time, as noted above, can be converted to a range and so the position of a scatterer relative to the sensor can only be determined via "receive ray" descriptor data. A model for the sensitive directions within the beam and the beam direction as a function of time is given in section 10.1.2.2. The lidar enterprise level-2 CMMD provides separate illumination beam and receiver patterns and orientations to support error analysis and be "bistatic ready" in the event an effectively bistatic lidar system emerges.

10.1.2.1 Illumination pattern

While the ranging accuracy of a lidar is not particularly sensitive to exact spatial illumination pattern knowledge, products dependent on the relative magnitude of spatial variations of returning radiance can not be of high quality without knowledge of the illuminating beam pattern. The illuminating pattern has basically two parameters: its direction (corresponding to a vector from the sensor along the peak of the illuminating beam, also called the beam axis) and its gain pattern, which may be factored into a normalized shape (that is, the radiant intensity function of direction about the beam axis where the response is unity) and the magnitude of gain on its axis at its peak[32]. We shall use unit vectors to specify directions and

[30] We consider devices that do non-time of flight ranging with laser illuminators to be "scanners", not lidars.

[31] The term, "receive beam" is equivalent to the instantaneous field of view of a sensor. The nomenclature is physically valid as it is derived from the principle of reciprocity. Also, it is more compact than "instantaneous field of view".

[32] The axis is defined to pass through the peak of a beam pattern. There is no implication of symmetry about the axis, although it is common that beams are rather symmetrical about their peaks.

orientation in an earth frame (ECEF) for the illuminating beam axis and the axis of the field of view of the receiving array (also referred to as the receive beam). We also use unit vectors to describe pattern directions about the illumination beam axis and to describe ray directions of the sensitivity of each detecting element in the focal plane array. Those unit vectors are given in terms of direction cosines relative to the plane axes (as opposed to beam axes[33]) of the beam and sensing array. The beam axes may be swept around as a function of time (scanning) and the look directions of individual detecting elements determine the rays along which the lidar makes range measurements.

An oriented pattern in space can be described in terms of two orthogonal vectors defining a plane. The axis of that pattern is in the direction of the cross product of the two basis vectors. So we consider a beam with a possibly elliptical power profile when viewed in a plane normal to its propagation. A circular beam is the degenerate case when the major and minor axes of an ellipse are of the same length. Define SC.Pulse.BmMajorAxis.*TileNum*(ip,j) to be an ECEF vector in the beam normal plane along the major axis. Similarly define SC.Pulse.BmMinorAxis.*TileNum*(ip,j) to be an ECEF vector in the beam normal plane along the minor axis. If the beam axes are equal, these vectors are interchangeable, but still define the beam axis uniquely. Despite the fact that most lidars will steer using beam optical components such as mirrors and keep the internal beam and receiving arrays relatively stable in the platform frame of reference, we will treat the impact of the adjustment of the transmit and receive beam directions as equivalent to rotating the beam or array plane about a sensor reference point. The SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j) (represented by $\mathbf{u_{maj}}$ in figure 10.1.2-2) and SC.Pulse.BmMinorAxis.*TileNum*(il,ip,j) (represented by $\mathbf{u_{min}}$ in figure 10.1.2-2) vectors are the *effective*[34] earth frame orientations of the beam plane. The index, ip, appearing in the beam axis data arrays ranges over slow time in pulse indexed samples. See figure 10.1.2-1.

Further, define a right handed system with the beam or array axis in the direction of $\mathbf{u_{min}} \times \mathbf{u_{maj}}$. Then any unit vector, **r**, in the outward hemisphere containing the beam axis can be specified by its components, the direction cosines $\cos\alpha$ and $\cos\beta$. See Figure 10.1.2-2. These **r** vectors are the directions of "rays[35]" for which the lidar sensor collects data determining round

[33] This choice was made so that we can convey not only the direction of the optical axis, but the angle about this vector through which an axis in the focal plane is rotated. In other words, we give a 3-D orientation of the Tx and Rx beam patterns.

[34] "Effective" geometrical quantities are equivalent to actual geometrical quantities in terms of effect on measurements, but may be simpler in the sense that single straight rays replace any multi-segmented paths with different directions in each segment.

[35] Strictly speaking, there are illumination rays and there are sensor receiving rays. There is no guarantee that a single photon remains on a receiving ray that is congruent with its illumination ray. In fact, unless the precision of the processor is quite coarse, these will not generally match, and we would never know for sure if they, in fact, happen to.

trip photon travel time. This model for directions in the beam or in look directions of array elements uses the direction cosines to specify directions, and the illumination pattern, SC.Laser.Beam(il,i,j) is given as a function of cosα and cosβ. The indices i and j of the beam are integer steps of size SC.Swath.DelCosAlpha and SC.Swath.DelCosBeta.

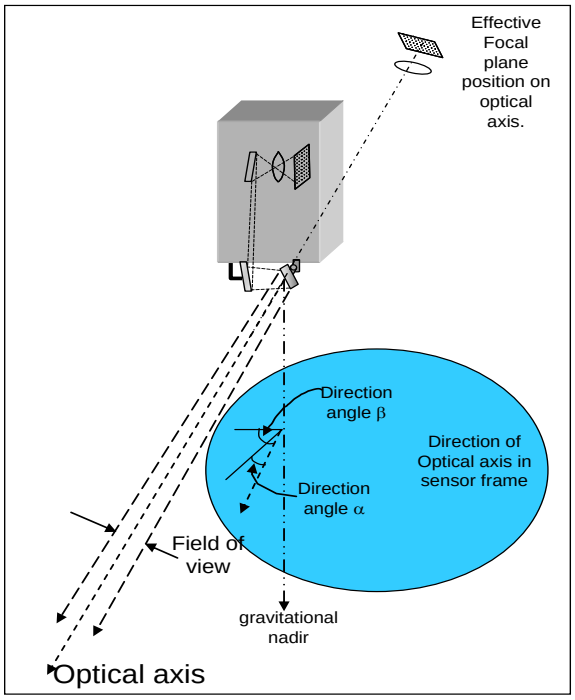


Figure 10.1.2- 1 Effective Position and Orientation of Array and beam Planes
Note that the principle of working in terms of an effective sensor position and orientation is independent of the configuration of the optical bench, one concept of which is illustrated in the figure.

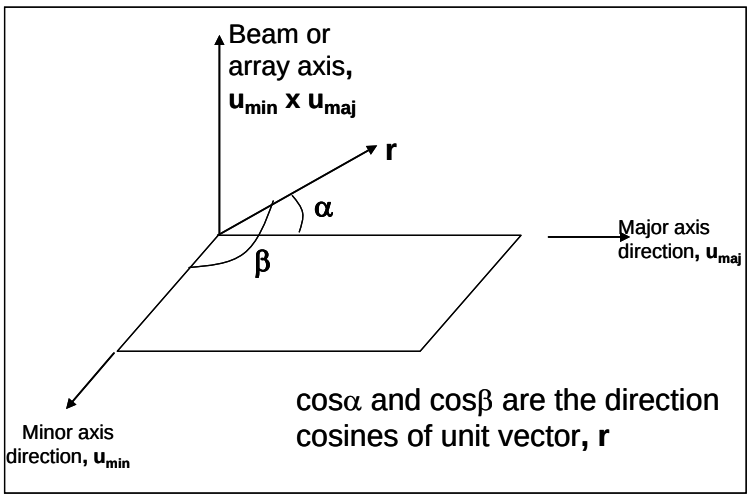


Figure 10.1.2- 2 Direction cosines relative to beam and array planes

10.1.2.2 Receive pattern, Axis and Ray directions

The focal plane is associated with a physical surface of sensitive elements and is thus somewhat more tangible than the beam normal plane, which at most is associated with the effective exit aperture for the laser output. We use the same procedure, defining two axes in ECEF to orient the array around its effective normal/look axis. Again, the *effective* direction that the array faces is that which incorporates the impact of steering elements in the optical chain of the sensor. Define SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) to be an ECEF vector in the focal plane along the major axis. Similarly define SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) to be an ECEF vector in the focal plane along the minor axis. If the receive beam axes are equal (square array), these vectors are interchangeable, but still define the Array 'look' axis uniquely. Despite the fact that most lidars will steer using optical components such as mirrors and keep the beam and receiving arrays relatively stable in the platform frame of reference, we will treat the impact of the adjustment of the mirror directions as equivalent to rotating the beam or array plane about a sensor reference point[36]. The SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) ($\mathbf{u_{maj}}$ in figure 10.1.2-2) and SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) ($\mathbf{u_{min}}$ in figure 10.1.2-2) vectors are the *effective* orientations of the focal plane axes. Define a right handed system with the sensing array optical axis in the direction of $\mathbf{u_{min}} \times \mathbf{u_{maj}}$. Then any unit vector, **r**, in the outward hemisphere containing the array axis can be specified by its components, the direction cosines $\cos\alpha$ and $\cos\beta$. This model for directions in the look directions of array elements uses the direction cosines to specify orientations. We create a list of direction cosine pairs for each and every element in the focal plane capable of producing lidar data called SC.Array.RayDirs(ia,ie,j), where ia is an array index, ie is an element index, j =1 for the $\cos\alpha$ and j=2 for the $\cos\beta$. This is a list, not a grid of data, so unlike the illuminating beam case, we have no steps in the direction cosines to specify. The element index advances along the rows of the sensor elements so that if the array is 10rows x 5 columns, the element at (row2, column3) has index (2-1)x5 +3=8. The ray directions in SC.Array.RayDirs(ia,ie,j) are given as unit vectors relative to the focal plane axes via direction cosines. The focal plane axes are given in ECEF on a pulse by pulse basis. Thus, the ray direction unit vectors can be directly evaluated in ECEF by applying the unit vector constraint to obtain the out of focal plane component from the ray components along the major and minor axis of the array given by SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) and SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j).

10.1.2.3 Optical axis direction, Steering of Tx and Rx beams

The variation of the illumination and receive beam axes in time is provided via the pulse indexing of the major and minor axis direction unit vectors. That is, each has an index, ip, which

[36] This metadata definition also applies to lidar designs that move the sensor head mechanically.

ranges over all of the pulses and thus is a sequential list of orientations at the times of each pulse. The variations are due to platform/sensor movement as well as intentional scanning of the lidar to cover ground beyond the immediate nadir FOV of the sensor. The instantaneous ECEFdirection of a ray from an individual sensing element on the focal plane is given in terms of the directions cosines from pulse indexed ECEF focal plane axis directions. So, for purposes of computing ranges and XYZ positions, the pulse indexed ECEF directions of focal plane axes and the receive beam pattern are all of the geometric data that is necessary. [Roundtrip times and a time to range model are also needed for XYZ point computation.]

The error propagation model (see 10.1.5) requires data on the uncertainty in rotations about the line of sight, which we also call the optical axis. The line of sight is derived from the two focal plane reference directions. The angle, gamma, is measured (counterclockwise) about the optical axis (looking towards the sensor) and is an angle between two planes that are orthogonal to the focal plane array. However, all that enters the error model is the uncertainty (that is a variation) about the mean of the orientation of the focal plane about the optical axis. Thus, an absolute reference is not required, as it will be subracted out in the process of developing the error covariance data.

Parameterization of the steering of the lidar look direction relative to the platform is another matter and is discussed in the next section.

10.1.2.4 Scan Parameterization

(see 10.1.4.2 for discussion of scan as part of the measurement hierarchy)

While the instantaneous beam axis directions do permit the computation of the direction of rays for each sensing element on the focal plane, the strategy of using periodic beam axis motion to sweep out a mapping volume has impacts that should be accounted for either in the sensor recording and control system or in the processing. For a sinusoidally varying scan direction, there is a region within about 20 degrees of phase zero and phase 180 where the sweep rate stays within 6% of a constant uniform value. Over such a range, compensation for the impact of beam scanning might be rather simple. Depending on the sophistication of the sensor control and, the processor and, the degree to which area collection requirements push for a high duty cycle of collection op time, the knowledge of where each pulse is in the periodic scanning cycles is of potentially great importance. Therefore the beam model includes a parameterization of the scanning action of the sensor beam control mechanism.

Knowledge of simple flags such as whether a pulse was a scan turn around pulse or not, whether the scan direction is + or -, and how far off nadir the beam is for a pulse are not good choices for scan description in the presence of pulse level beam orientation data. Even in the absence of pulse level beam orientation information, such items require additional assumptions or constraints to be meaningful. Furthermore, these parameters do not support assessment of scan

dependent geolocation error or distortion in a quantitative sense. So this model dispenses with those items altogether and introduces a more general lidar scan parameterization.

Scans that have been used in lidar systems include linear scans at various orientations to platform velocity or local vertical and, conical scans. The latter have the property that an entire scan's worth of pulse collection can be at a constant angle off nadir. Linear scans can be considered the limit of a general elliptical scan "cone" when one axis of the scan in angle space is reduced to zero. Consequently, a single parameterization of beam scanning as lying on a general elliptical cone serves to support linear, circular conical and asymmetrical conical scanning of the lidar merely by adjusting *values* of a few basic parameters. Linear scans are likely to be intended to be in cross track or along track directions. In principle there is no reason to limit the angle between scan direction and ground track to only those. The scan may make an angle to the platform trajectory that is neither 0 nor 90 degrees. The idea of a scan is that it is a repeated pattern, used many times in a lidar collection. A scan is the direct consequence of angular beam axis control and is better parameterized in terms of such sensor frame angular variables than, for example, spatial variables on a nominal ground surface. Thus we specify a scan by the following parameters:

- Scan will be an elliptical cone possessing-
    - Angular major axis extent, SC.Scan.Axes.Theta.Maj
    - Angular minor axis extent, SC.Scan.Axes.Theta.Min
    - Direction of scan cone axis, $\mathbf{u}_{\mathbf{scan_cone}}$, a unit vector fixed (during a single scan) with respect to the sensor frame, given by direction cosines of α_{scan_cone}, β_{scan_cone} from the sensor frame x and y axes. The data tag is SC.Scan.U_Cone(iscp, ich,k). The index of individual values of the unit vector scan cone axis direction threads to the pulse number via SC.Scan.ConePulseTable(isci), a list of pulse numbers which are the first pulses of new scan cone directions. All scans occurring during pulses after the last pulse number in the SC.Scan.ConePulsetable(isci) list use the last scan cone axis direction in SC.Scan.U_Cone(iscp, ich,k) . In particular, if a collect uses only one scan cone axis direction, only pulse zero will be in the SC.Scan.ConePulseTable(isci) list and there is only one value for the cone axis in SC.Scan.ConePulsetable(j).
    - Orientation of the scan major axis. A plane cutting the scan cone on the major axis and containing the sensor position determines a triangle in a plane fixed to the sensor body. The vertex angle of the triangle is the angular size of the scan major axis, SC.Scan.Axes.Theta.Maj. Specifying the normal to this plane fixes its orientation. The normal may be parameterized by a pair of direction cosines in the sensor frame. Giving the ECEF components of the axes of the sensor frame from which the direction angles are measured permits the scan major axis plane to be oriented in the earth frame as well. [See, for example, SC.Pulse.SensorAxisX.*TileNum*(ip,j)] The normal to the plane of the scan major axis, $\mathbf{n}_{\mathbf{scan_major}}$ (data item SC.Scan.N_Major(iscp, ich,k)) is given via direction cosines α_{major} and β_{major} from the sensor frame x and y axes (see Figure 10.1.2-3

below). The values of $\mathbf{n}_{\mathbf{scan_major}}$ for two adjacent scans, along with the corresponding sensor frame x and y axis ECEF vectors and position data can be used to determine the expected volume of overlap for two scans and their nominal alignment.

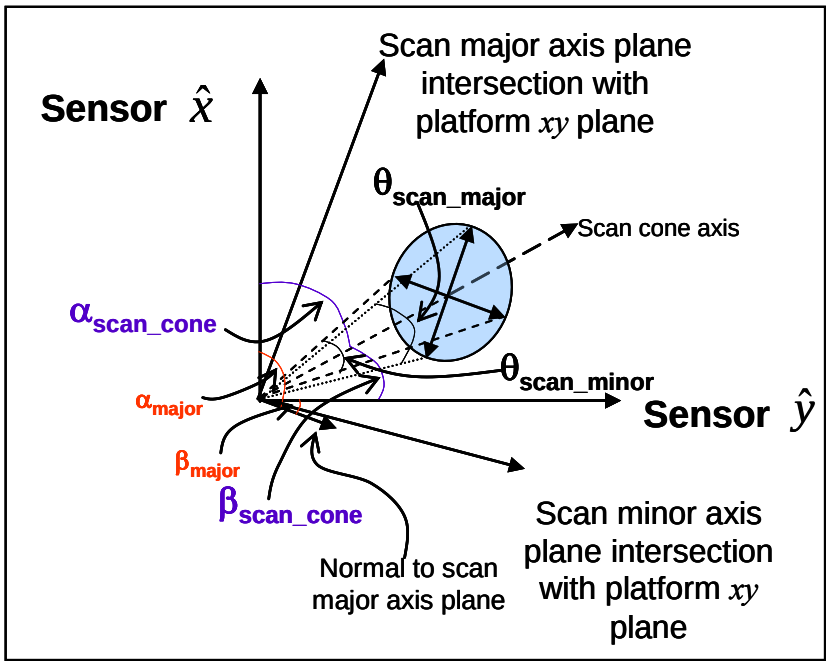


Figure 10.1.2- 3 Scan Cone Axis described by Direction cosines.
The scan cone is generally elliptical with major and minor angular axes of size θ_{scan_major} and θ_{scan_minor}. The cross product of $\hat{x}$ and $\hat{y}$ is in the hemisphere containing the scan cone axis look direction so the figure should be interpreted as the scan cone being directed behind the plane of the diagram.

Note that since the direction of the scan axis is given in the sensor frame, if the sensor is mounted with its x axis along the ECEF velocity vector, and the major axis plane contains the y-axis, then the forward motion of the sensor causes the scan traces on the ground to appear as a zig-zag pattern with scans inclined with respect to the ground track. If it is truly desired that the scans be orthogonal to the ground track, appropriately matched cone axis offset jumps and scan axis direction changes have to be computed for the specific ECEF velocity.

Note also that the scan parameters do not give the orientation of the sensor array about an optical axis; they only determine the direction of that axis. The scan parameters given also govern both the illumination and receiving axes, although at the most detailed level, supported by pulse data, not scan parameters, these optical axes may need to be slightly different to account for long range to the target and motion effects such as light aberration.

All scans are assumed to be periodic, implying repeated beam articulations with a single underlying phase, ϕ, as a function of pulse supplied in a data table for each collection operation. For a given pulse, the beam angle from the cone axis along the major axis direction is given by

$$\theta maj = \theta \max_{maj} \cos\phi(pulse_number)$$

and, along the minor axis direction it is given by

$$\theta \min = \theta \max_{\min} \sin\phi(pulse_number).$$

Thus, phase equals zero is defined as the beam being directed to the largest value of θ_{maj} in the scan major axis plane and θ_{min} in the orthogonal plane. This occurs when the beam direction component in the direction given by $\mathbf{n}_{\mathbf{scan_major}} \times \mathbf{u}_{\mathbf{scan_cone}}$ is at its maximum (refer to Figure 10.1.2-3).

Directing the sensor optical axis along this parameterized elliptical cone requires computing direction cosines for the optical axis from the values of θmaj and θmin obtained from the scan phase for each pulse. This is readily obtained by inspection of figure 10.1.2-4.

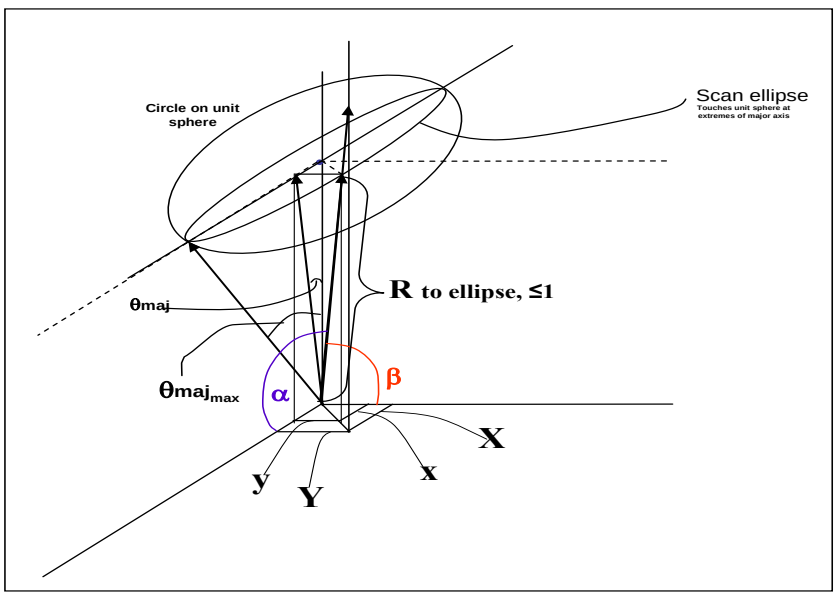


Figure 10.1.2- 4 Scanning on an elliptical cone parameterized by scan phase.

$$\cos^2\alpha = \frac{\tan^2\theta maj}{1+\tan^2\theta maj+\tan^2\theta\min} \qquad \cos^2\beta = \frac{\tan^2\theta\min}{1+\tan^2\theta maj+\tan^2\theta\min}$$

The required scan phase for each pulse will be found in data tag SC.Pulse.Scan.Phase.*TileNum*(ip, j). In populated scan phase tables, it is permitted that less than 2π of phase appear before the scan cone axis direction might be reset by the

SC.Scan.U_Cone(iscp, ich,k) and SC.Scan.ConePulseTable(j) data. Resetting after only a π scan phase change[37] would be typical of linear scan operation.

- Scans are also normally parallel within a collection operation in the sense that the major axis plane normal (which is referenced to the sensor frame) is likely to be fixed for a full swath collection[38]. However, the parameterization can permit a more general case without adding data volume in the expected case of just one scan major axis direction for a collection operation. So the data tag, SC.Scan.N_Major(iscp, ich,k) permits the scan direction to be modified anytime the scan cone axis is modified. All scans occurring during pulses after the last pulse number in the SC.Scan.N_Major(iscp, ich,k) list use the last scan major axis direction in SC.Scan.N_Major (j,k). In particular, if a collect uses only one scan major axis direction, only a pulse zero entry will be in the SC.Scan.N_Major (j,k) list and there is only one value for the scan major axis. The "j" index is that used in the "first pulse of new cone direction" list SC.Scan.ConePulsetable(j). Thus, if the second cone direction begins on the "ipth" pulse in SC.Scan.ConePulseTable(j), the second scan direction, if there is one, does too.[39]

The Limiting cases of interest:

- The scan elliptical cone can have equal axes and thus be a (circular) cone in the sensor frame, and
- The scan elliptical cone can have a minor axis of zero, corresponding to a plane in the sensor frame. The scan ground tracks for a flat ground facet would then be linear (presuming no acceleration of the platform).
- A circular sensor frame cone with a non- nadir scan cone axis would intersect a flat ground facet in an ellipse. If the orientation of the scan cone axis is normal to the ground (typical of Nadir orientation), and the major and minor axes are equal, then the arbitrary choice of defining the major axis to be cross track can be made without affecting the accuracy of the description of the scan pattern. The only pulse dependent parameters for scan that are properly defined are the scan phase and scan count [but there may be many pulses before a scan counter increments].

[37] Scan phase can repeatedly run in pulse strings that go only from 0 to pi as well. This is a "carriage return" style of scan rather than a zig zag with a back scan.

[38] Platform roll, pitch and yaw with respect to the velocity vector can lead to rotations/distortions of the scan cone as viewed in the earth frame.

[39] This scheme permits a rather accurate, but not exact, representation of a Lissajous scan pattern as well.

We nominally characterize the scans in a collection via high level descriptions that do not have the precision of the scan phase driven parameterization and these parameters are suitable for search and discovery purposes. The appropriate value is provided in SC.Scan.Type.

See subswath discussion in 10.1.4.3 for scan overlap description.

10.1.3 Channelization Model

10.1.3.1 Definition of lidar Channels

What follows is a major elaboration over the basic definition of channels and channelization given in section 5.1.

Enterprise Level-1 (raw channelization)

At enterprise level-1, a lidar channel is associated with each separate fast time sampled measurement set of a separate raw data stream of IPP based lidar data. Each such raw data stream of lidar data represents a single phenomenology that, in turn, corresponds to a specific *combination* of constraints on the physical degrees of freedom of the incident photon flux such as spectral band, polarization, field of view, angle of view to target field, modulation pattern, ... By "combination", we intend, for example, that one wavelength and one polarization can be in one channel's data stream and the same wavelength and a different polarization would have to be in a second channel's data stream. A second wavelength and the first polarization selection of data would be in a third channel. So a channel basically corresponds to a unique pairing of a laser illuminator and a detecting array, or "filter wheel"[40] position setting. Thus, a system with a single laser and focal plane detector array is a one channel system if it has no "filter wheel". If it has a filter wheel that operates such that every odd pulse is polarized along the focal plane array rows[41] and every even pulse is polarized in the cross-row direction, the odd pulses collected by this system constitute one polarization channel and the even pulses constitute

[40] Quotes are used around "filter wheel" to indicate that this covers all implementations of time dependent filtering of the photons being detected on a single physical array. The system may physically not use a rotating filter wheel, but rather a scanning mirror in the optical bench. Regardless, the channelization is implemented by temporal masking of the samples recorded off of a single detecting array. The data from the various channels is sequentially placed in the raw data stream and the optical chain acts as if it contains a "logical" filter wheel.

[41] Polarization channels are only given by the receive electric vector sensitivitiy alone if the transmitted photons are randomly polarized with no net beam polarization. When the illuminator is polarized, channels would need to be designated by both the Tx and Rx orientations. These may be later processed to VV,HH, VH,HV with respect to a plane of incidence, when it exists.

a second polarization channel. Similarly, if one laser-detector array pair looks at an offset angle relative to the vertical plane described by nadir and the instantaneous sensor trajectory described by angle α and a second separate laser-detector pair looks at an offset angle relative to the vertical plane described by nadir and the instantaneous sensor trajectory described by angle β, these are two channels.

- A system with two lasers of different wavelengths and a single detector array with spectral response wide enough to detect photons from each laser could constitute a 2 channel system if the knowledge of which laser was responsible for each measurement is tracked.
- If each of the two lasers were used with a separate dedicated focal plane detector array, that would also be a two channel lidar. It could also be regarded as two separate (near) simultaneously operated lidars. However, see correlation discussion below.
- If one laser illuminator is used, but two focal plane detector arrays are used (e.g., one for elastic scattering and the other for non-elastic photon scattering), that would also be a 2 channel system.
- A system with three orientation polarization filters and one laser is a 3-channel system.[42]
- A system that uses polarization pulse compression with a rotating E-vector and 2 detector arrays with orthogonal polarization filters is a two channel system.
- Two arrays might be interleaved on a single chip and have different spectral or polarization response. Even if the fastime sample stream is multiplexed across both arrays, the data from each individual array belongs to a separate channel.

  The objectives of channelization might be to:
    - collect multi-wavelength data for spectral diversity
    - collect multi-polarization data for polarimetric lidar
    - Collect data at higher area rates by using field of view diversity of simultaneously collecting channels

The physical instrument might collect a single channel level-1 data stream as two (or more) physical data streams at level-0. In-sensor processing, such as taking two chips (one physical array each) hard mounted on the focal plane with complimentary instantaneous FOV and responding to the same pulse can have their data streams merged into a single effective stream. The sensor outputs this as a single channel. It truly is a single channel because the sensor does all of the packetizing as if this were a single array source. This is presented as single channel data at level-1.

[42] It is possible to operate a 3 channel system in a mode that collects data in only one channel. The collected data set is then a single channel data set.

Another single channel case with multiple level-0 channels can arise with post sensor processing of the same two chips whose data was kept as two data streams in the sensor and sensor comms (even if eventually multiplexed for transmission). While combining these streams could make this data level-2, even if it is logically equivalent to what is described above as in-sensor processing output, the level-2 status is not necessarily the case. It depends on who does the processing and how it was done. It is appropriate that post sensor processing perform some adaptive correction if there is overlap provided. In that case we have unarguably moved to level-2, no matter who performs the processing. But the issue of reformatting the data without modification into a single channel stream as was done by the sensor in the in-sensor case is subtle. Why should we consider on-board reformatting to create one L-1 channel while off board reformatting creates an L-2 channel? The reformatting is not processing, per se. So, if the vendor does the reformatting off board and delivers the data as L-1, then that is the vendor's *implementation* prerogative. The vendor is warranting that the data has not been processed but rather the single channel data has been assembled for further processing, per the intended design of the sensor. However, if a second party performs the same operation without the "vendor's" warranty that this is still a single raw channel, then **we must assume** that the sensor operator vendor is a separate entity who captured the data in separate streams and delivered them without reformatting because they were deemed to be fundamentally separate and perhaps not properly suited to merging without some adaptive or deterministic corrections. The assumption may be incorrect and the vendor may have just been lazy, but the definition of channelization will have to attribute due diligence to the level-1 formation process.

The mere fact of having a single laser per array composite element vice two does not determine the channelization at level-1. If the sensor vendor/operator delivers separate channels at level-1 they are separate. If they deliver a single channel, they are a single, if merged channel. The vendor may wish to be careful regarding being too sloppy about just merging when two lasers are in use. It is very difficult to guarantee simultaneous flash times so that the two lasers act as one. If a separate data stream describing the flash times is needed to compute round trip times for the separate chunks, the vendor is safer presenting these as separate channels. The quality of a poorly merged channel is not good. The quality issue is finessed at enterprise level-1 by keeping the data streams as separate channels. This way the impact to quality has been shifted to the L-2 creation process.

> If the parts of a composite array contribute to a sensor merged channel data stream, they still may warrant array chunk data description. See SC.Array.Chunks(ia,ich,j) and SC.Array.LatticeDesc(ia,j).
>
> For enterprise level-1, the description of the individual channels in terms of laser, array and filter choices is provided in SC.Channlzn.Descr(j), where j is a row index for each current product channel.

Note that a channel of data carrying the sensor position or attitude as a function of time is not a *lidar* channel. Also, if multiple lasers are used, each channel may have its own time sequence for pulse transmission times rather than a common pulse transmission sequence for the lidar system. This could be true even if there is only a single pulse transmission command sequence for the full lidar system. The pulse unique delays of each laser create a channel specific transmission time sequence.

Multiple Channels or Multiple Lidars?

The technical distinction between two channels of one lidar and having two lidars is that the channels of a multichannel lidar are presumed to have negligible decorrelation of errors in both the sensor position and attitude and consequently share a common set of error propagation data in the lidar model. Also, a single pulse is assigned to a single scan for a single lidar. So, multiple channels have the same scan to pulse assignment if they are in the same lidar. While spectral or polarization channels will generally share the same scan cone parameters leading to simultaneous sampling of the same scene regions, spatial channelization can result in the same scan index having different scan cone parameters, leading to different scene coverage for two channels within the same scan in one lidar. This drives us to add a channel index, ich, to those scan parameters that need to be given potentially channel unique values. Of course, the situation of there only being a single channel, or all channels sharing scan parameter values become simple special cases of data population with respect to ich. Note well: The meaning of the index, "ich", is given through channelization description data such as SC.Channlzn.Descr(j,k). For the scan parameterization to work properly, it will be necessary for the channel index range to extend to the collection channels in any data set with multiple channel specific scan parameter values.

On the other hand, two separate lidars are to be presumed to have uncorrelated geometric errors and each lidar would generally need to have its own error propagation data (if population of error propagation data is a system requirement). Two separate lidars also *may* have have distinct assignments of scans to pulses. That is, if at the same time, two or more scans are assigned to a single pulse index value, then two or more lidars are in operation. We do not (and can not) have a multichannel lidar that happens to be running different scan sequences simultaneously in each channel. This will be more clear after reading the discussion of scans in 10.1.4.2.

When a system's intended exploitation conops is independent of the issue of geometric error correlation between channels, system operators may either choose to represent effectively two lidars as a two channel lidar or to prepare that data at level-1 as two level-1 files and the consequence of combinations of both lidar data streams could be put off until level-4. However, per the definitions within this CMMD, users are expected to be able to

safely presume geometric error correlation for multiple channels, but not for multiple lidars. The level-1 data provider is making this implicit warranty in the channelization strategy of the delivered data.
One could anticipate that the correlation necessary to consider the system to be a multichannel lidar would result from a sharing of key physical components. A common steering mechanism, common detection plane, common optical bench components, or a common illuminator are likely design features leading to the multiple data streams coming off the sensor to qualify as channels of a single lidar.

Enterprise Level-2 and above (derived channels)
At Enterprise level-2 and above, the channelization may have been preserved as that of the level-1 data, or it may have been processed via channel combination of some kind into a new and likely reduced set of channels. The merging of point clouds from adjacent FOV channels is all that is necessary to combine channels of that type. This process may or may not entail registration or alignment. Creation of stokes vectors and special spectral signature channels involves more elaborate channel combination.

At enterprise level-2, the original collected channelization of the level-1 precursor will often be retained. However, it is expected that in some cases the level-2 product may contain a new channelization that may be more physically significant than those of the level-1 data. An example is converting native row and cross-row polarization data to vertical and horizontal polarization or to stokes vectors. Vertical polarization is when the E-vector is in the plane of incidence and horizontal polarization is when the E-vector is normal to the plane of incidence. Defining these polarizations requires a reference surface for the scattering scene. The vertical and horizontal polarization channels would be created by superposition of the native level-1 channels. We use the earth ellipsoid tangent plane for off nadir down looking lidar collection. It is not possible to characterize a channel in terms of a variable ground slope over the scene at level-1. At level-2, nominal polarizations or point specific polarizations can be computed. If the data is collected on scans that pass through vertical (0-nadir) collection polarization filters can still be used, but the plane of incidence reference fails. Vertical polarization might then be arbitrarily defined, for example, as the electric vector being in the plane of the optical axis and the focal plane direction listed as the major axis SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j). Arbitrary linear polarizations can be expressed as having vertical and horizontal components (when a plane of incidence is defined).

For enterprise level-2,3,4, the description of the individual channels in terms of precursor channels, often just the collection (level-1) channels, is provided in PF.Channlzn.Descr.*m*(j), PF.Channlzn.Disp.*m*(j), PF.Channlzn.Hist.ChTrace.*m*(j,k), PF.Channlzn.Hist.Comb.Code.*m*(j), PF.Channlzn.Hist.Comb.Func.*m*(k), and PF.Channlzn.Hist.Comb.Alg.*m*(k). In these items, the evaluated substring, *m*, supports a

history dimension to the data, providing information about channelization in all previous processing stages that resulted in creating the channelization for the current data set, even if earlier channelizations were transient interim data states. The range of m, the integer conversion of the numeric string, *m*, is the same for each of these PF.Channlzn.* items. Were it to be the case that PF.Channlzn.Hist.Comb.Func.*m*(k), or PF.Channlzn.Hist.Comb.Alg.*m*(k) was not populated in earlier data sets and the disposition as indicated at earlier stage outputs did not anticipate the later population of those items, when those items do eventually occur later in the processing chain outputs, they must be populated with an internally consistent range of *m* within the data set that provides them. For example, PF.Channlzn.Hist.ChTrace.*m*(j,k) and PF.Channlzn.Descr.*m*(j) refer to the same channelization when the values of *m* are the same. Thus, even if PF.Channlzn.Hist.Comb.Func.*m*(k) first appears at level-3, its range of *m* will need to include channelizations that may have been documented by populating only PF.Channlzn.Hist.Comb.Alg.*m*(k) in the first level-2 product. So the top priority is consistent use of the channelization index, m, within a single product/ data set. Restricting channelization incrementation to only the occasions of a channel combination stage is not required, and may actually leave some uncertainty about interim stages of channel combination. So the processors have the freedom to define interim stages and a channelization should be assigned to each. A stage that involves channel combination will see the channelization increment just as a stage that does not involve channel combination. The description of the channelization at that stage would document whether or not a channel combination had been involved to change things or not. Vendors ought to take care to count any stage with a channel combination process as a stage that generates an "m-layer" to the channelization history, but interim processing stages that do not affect channels and that are not output as CMMD compliant data sets, can be left out of the counting without negative impact. However, if the stage is significant for a non-channel related issue, it will need to be counted and the channelization that goes with it will be a repeat of the previous stage. (Yes, channelization 2 could be the same as channelization 3, but some other significant change, such as internal registrations may have been performed that preserves the channelization properties, just not the channelization count.) This is particularly the case when an enterprise L-2 or L-3 product has been created.

10.1.3.2 Disposition of lidar Channels

This CMMD provides metadata that describe the intended use of channels in the current file. The principle options here are either intermediate use or direct exploitation of channels that have been prepared to be either dominantly signal, dominantly clutter/noise, or a mix with significant levels of signal and clutter/noise. Intermediate channelization could be intended for:

i. spectral or polar fusion step
ii. spatial fusion step

iii. temporal fusion step

See PF.Channlzn.Disp(j,m) .

10.1.3.3 Describing Lidar Channel History in Metadata

This applies only to enterprise level-2,3 and 4 data. The following elements constitute the history of channels in a data set at a level above enterprise level-1.

a. Channel n of current product derived from channels a,b,c… of parent
b. Combining process to arrive at channel n
    i. Spatial juxtaposition
        1. Point file merge without adjustment
        2. Point file merge with adjustment (registeration, alignment)
    ii. Linear superposition with weights w_i.
    iii. RSS (possibly weighted)
    iv. More general function expressible by a simple equation $Cn=f(c_1,c_2,c_3,\ldots)$
    v. A complicated algorithm with conditional, likely data dependent steps for combination. Such an algorithm needs to have a registered name [could be listed in the CMMD] which can be included as metadata to support documentation of how the data has been processed.

Data items for documentation of channel combination history are:

PF.Channlzn.Hist.ChTrace.*m*(j,k)
PF.Channlzn.Hist.Comb.Code.*m*(j)
PF.Channlzn.Hist.Comb.Func.*m*(k)
PF.Channlzn.Hist.Comb.Alg.*m*(k)

10.1.3.4 Indication of Lidar Channels in Metadata

Unless dedicated tiles are used for single channels, we need to be able to indicate the channel for each lidar point and waveform clip. This is done via incorporating the channel number into the data tags LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) and LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) at enterprise level-2. For waveform data at level-1 and level-2, ES.WFClp.ChNum.*TileNum*(ipi,iray) is used.

Consider the case that arises when channels need to be spatially merged to form a contiguous block of output point cloud volume. This is likely due to collecting sub-scan strips on separate arrays in separate channels at level-1. After processing, there is virtually no exploitation based reason to carry the channel provenance. An anomaly resolution scenario might exercise channel knowledge, but the channelization in this case is just an engineering approach to improve collection rate with available sensor components. This is distinct from channelization to support increasing the number of observable parameters. If only area FOV diversity coverage channelization is all there was to the collect, it makes sense not to carry the channelization past L2. A single channel with the identification of the parent channels and merge process heritage is

all we would track into subsequent products. There is a reason to be able to clump semi-scan[43] channels. The positional errors should always be highly correlated for a single channel within a single frame. The errors between channels may be different and require alignments or registrations. If we have lost the channel assignments before performing such operations, the opportunity to do them is pretty much lost forever. If we have assigned points to tiles by channel, we retain the identification, but unless the tiles are rather small, there will be bands of blank space in each (single channel) tile when displayed as a point cloud. We would have to figure out the adjacency of point sequences within tile pairs and extract sub-scan strips for registrations and alignments. That would be easy if there is a scan number we can assign based on the IPP number on a point record. Display tools can create a defacto merging of such channels merely by enforcing consistent point positioning in the projection of the data to the viewing screen. However, such tools may indicate any voids or redundantly collected regions (probably strip-like) with darker or brighter regions on the display. That is, channel combination in a display tool alone may not make expected corrections in coverage when combining local FOV diversity channels.

No added metadata items are needed if channel dedicated tiles are used. Mixing channels in a tile requires a channel indicator, however. For this, see LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) and LD.Point.Pos.*ChNum*.*TileNum*(ipt, j).

A subtle interplay between the channelization description scheme and the tiling scheme used in this CMMD exists so that the number of metadata items and dimensions of data arrays can be minimized. Pulse oriented tiles need to be constrained to a single channel when used to support documentation of IPP sequences that are distinct for different channels. This is done via the dedicated constraint construct for tiles (explained in 10.2.1) and a single value of *TileNum* for pulse oriented tiles can only apply to a single channel. Channels due to 2 lasers and one array will need separate IPP sequences. Channels using independent laser and array pairs might need separate IPP sequence data for SC.Pulse.TxCount.*Il*.*TileNum*(ip,j), but for best accuracy, should supply at least the SC.Pulse.TxDelay.*TileNum*(ip,il) data. Channels that use separate arrays or filter wheels may have a common IPP sequence. We don't need to populate pulse tiles redundantly when the same IPP sequence is used for multiple channels. When there is only one pulse sequence in a data set, but there are multiple channels, the single IPP sequence provided should be used for all channels. An error condition could be detected by determining whether there are more lasers in the channels than arrays when there is only one IPP sequence. Such a case would mean that the data set should fail a validity test.

[43] Meaning each channel in a spatial diversity collection mode sees only part of the total collected scene.

10.1.4 Sensor Measurements, Hierarchical Organization (PCD, Scan, Subswath and Swath)

In this section we discuss a hierarchical ordering of lidar data based on a combination of natural collection and processing aggregations of individual lidar measurements. Superficially, we collect basic measurements into logical sequences of lidar frames (a pulse/IPP's worth of data each) associated with beam movements called "scans". In turn, we logically collect the scans into "subswaths", either because the collection scheme naturally had related scans in temporal blocks or because the processing naturally operates on contiguous scan groups, and finally, all the subswaths taken as a single collection form the "swath" or logical product-level aggregation of measurements associated with a single collection operation (often called a "flight line[44]"). Physical aggregations of data called "tiles" lump data into I/O oriented groupings sized according to natural usage volume capacity or H/W performance/addressing constraints. These relationships are summarized in the overview diagram Figure 10.1.4-1 and explained in greater detail in the following paragraphs of 10.1.4.

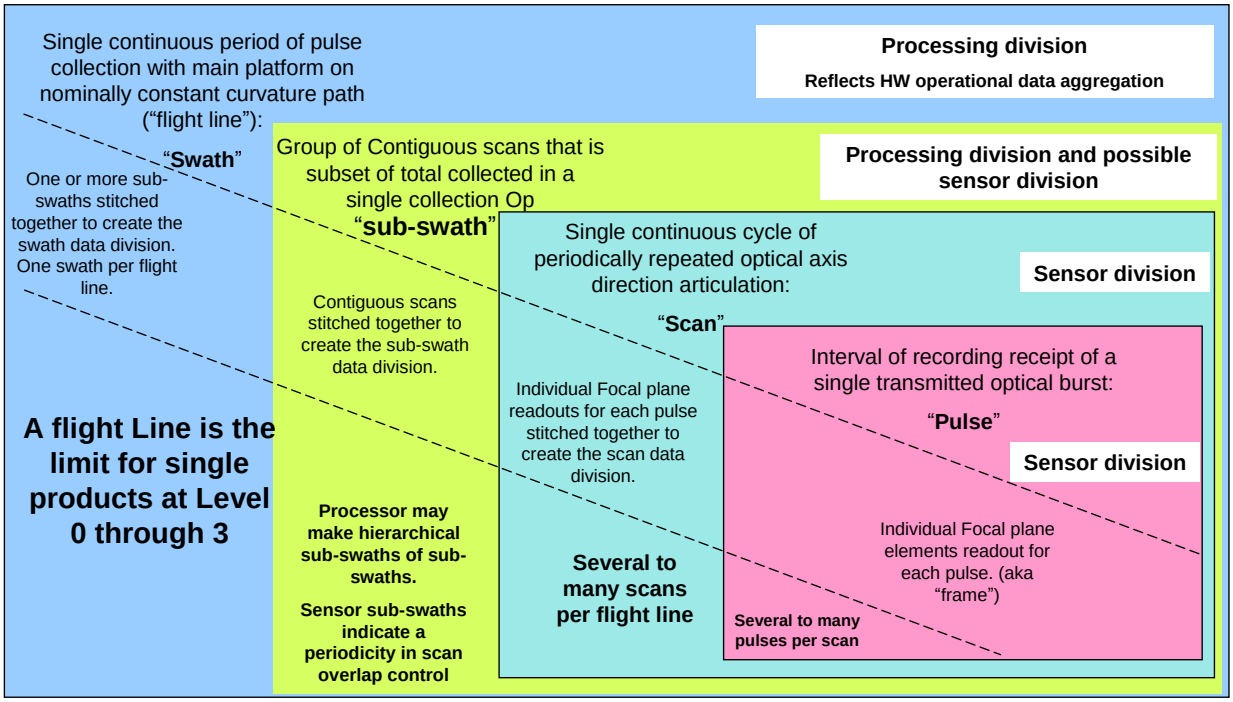


Figure 10.1.4- 1 Hierarchical relationship between pulses, scans, subswaths and the product level swath for lidar data sets

[44] The term "flight line" is clearly inspired by airborne lidar collection in a mapping-like application. While we may use this term in discussion within this document, the data model uses the somewhat more generic term "swath" for the product level full aggregation of collected data from a single collection operation.

10.1.4.1 Basic lidar Measurements

Technically, a single look direction (ray) and time of measurement[45] parameterize the instantaneous position within a 3-D scene probed by a lidar. That is, the lidar points are truly *measurement* positions. However, if there were not something to reflect light back to the lidar receiver at a measurement position, a null measurement would likely occur. In linear mode, a lack of returning photons results in only a noise level flux measurement at the time corresponding to a roundtrip for photons on a ray from sensor to measurement position and back. Ideally, in Geiger mode, no time is reported after quenching the array until a photon arrives and triggers an avalanche[46]. If there are no scatterers in the range swath for a ray, the sensor will usually not record a time and a null measurement is potentially recorded. However, there is always a finite probability that a noise event triggers an avalanche to create a spurious photon count. Until downstream processing, these spurious responses are indistinguishable from real photon counting events. Even later processing must use model based filters to statistically reduce the spurious photon counts. For lidars that operate using pulse compression, the raw return somewhat resembles that in linear mode. It also represents a time history of scene return, but time cannot be converted to range until the return has been "compressed" so that a single range maps to a single value of the parameter in the post compression domain, often an intermediate frequency or polarization value.

A lidar sensor collects data in temporally associated individual measurements by **p**ulse. The number of individual photon flux measurements or arrival times collected for a single pulse depends on how many sensing channels are recorded, how many individual reporting sensing elements there are on each channels's focal plane sensing component (generically an array, but not limited to a grid of APDs), which we designate as an **a**rray, and of course, the number of times per pulse that a recorded sample can be saved from each detector element. The sensor may expose and read out more than one array per received pulse. Each individual reporting sensing element has its own cone of optical sensitivity aligned in a look **d**irection (which we call a ray). So, there is a natural hierarchy of data organization by pulse (slow time of measurement), channel and array element ("spectral"[47] and spatial measurement grouping) that leads to a single direction's worth of time history measurement for every pulse. The most fundamental data

[45] In this statement, the "time of measurement" has two parts. The first part is the slow time, which pegs the sensor position. This position in turn, acts as the base of the look ray. The second part is the fast time, which is the basis for determining a range along the ray from the sensor to the probed scene point.

[46] Geiger mode is also susceptible to noise and spurious avalanche events due to thermal events and neighbor triggered responses (array element cross talk analogous to blooming in a video tube or CCD)

[47] In this sentence, the term "spectral" is loosely applied to spatial and non-space time diversity of signal parameters. This might be the wavelength band of sensitivity, the polarization orientation interval or even a look cone direction due to a partitioning of the field of view among focal plane components. It is recommended that spatial diversity be approached as a single logical array, even when multiple physical parts are reporting measurements from their elements. So, if two separately manufactured APD grids are mounted on the focal plane and report data during the same pulse interval they should (not must) be treated as if they are a single array.

division is the single **p**ulse, single **c**hannel, and single element **d**irection (PCD) time history. Thus, a single measurement point position in the scene space leads to a data parameterization in terms of a single value for each of the following:

- Pulse index
- Channel index
- Direction index within the channel's array

The collection of measurements over all directions for a single pulse is called a frame[48]. For a single quench per pulse Geiger mode lidar, a frame per pulse is a complete data record for a receive window. A single frame for a linear mode lidar has all of the returns in the instantaneous field of view, over the range swath for one pulse. This data could be processed in hardware to mimick the form of Geiger mode data by capturing only the first time that the photo currrent exceeds a particular threshold individually for each of the elements of a detector array. Such flash lidar data for a single pulse could also be termed a "frame". If the lidar is set to stare at a fixed scene position, rather than creating extended region "scans" with a frame sequence, a 3-D video clip (that's a 4-D array displayed frame by frame) would be the consequence.

For a single pulse, a PCD and a range derived from processing of the fast time signal from that PCD lead to a 4-D photon measurement "position" consisting of an XYZ position and a slow time. With a unique channel value assigned for each return, a level-2 point can be considered 5-D where the channel is the 5th dimension. While there may be as many ranges determined per pulse-detector as there are "quenches" in a Geiger mode, the number of unique ranges that can be determined in linear mode is limited by signature models and signal to noise. For pulse compression lidars, the time·bandwidth product of the modulation, duration of the receive window, signal to noise and processing all factor into determining how many independent ranges can be assigned as possible return measurements for a single PCD.

The PCD concept is illustrated by figure 10.1.4-2.

[48] A frame can capture all the channels if channelization is not using separate pulses for separate channels. So one array in one pulse would be a channel-frame, not merely a frame.

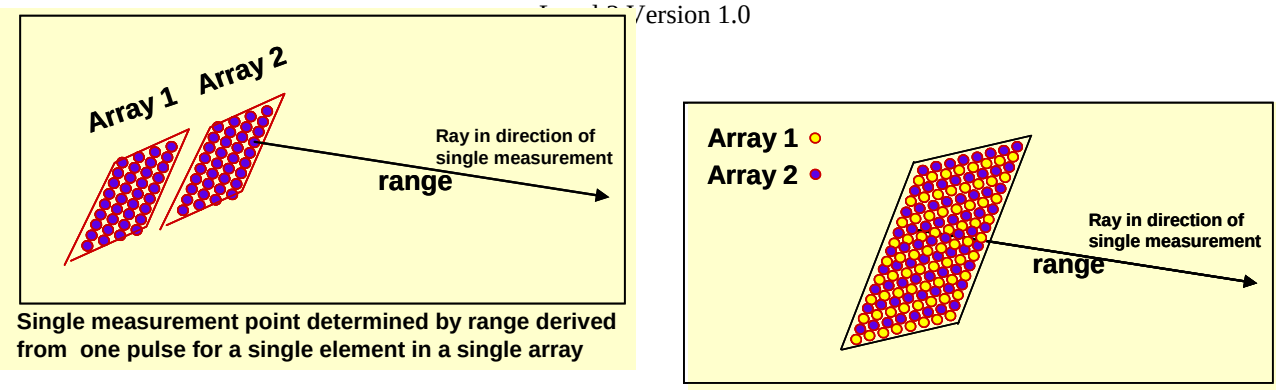


Figure 10.1.4- 2 PCD visualization

Typically, after undergoing several successive scattering events in the scene, the photon return is too low to rise above the noise threshold of a lidar. However, not all photons in a pulse on a ray undergo the same scattering sequence so that multiple return ranges are quite possible. The diversity of photon histories for a single ray-pulse depends on the size of the ray's sensitivity cone (resolution effect) and the density distribution of scatterers in the scene. This leads to the desire to be able to routinely code for 7 returns per pulse. This is accomplished by the coding scheme for ES.Point.RtnsCode.*TileNum(*ipt,j).

<u>PCD and the iray Index for CMMD Data Arrays</u>

This section explains the identification of an array and an element as a "ray" for identifying waveform clips. [But we can also match a point to the ray from which it was derived, when needed.] The purpose of the strategy described below is to minimize the data volume required to identify and find waveform clips. As indicated in the PCD discussion above, a ray is specified by the array on which a detector element lies and the specific detector element on that ray. From this, the look direction relative to the sensor frame can be determined by using the unit vector data in SC.Array.RayDirs.*Ia*(icase,ie,j). That can be converted to ECEF look vectors by incorporating the data specifying the effective array orientation in SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) and SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j).

There is little exploitation need to ever trace a point back to a specific element on a focal plane array except to understand the specific look direction for that point in support of finding waveform clips that represent a desired region of the scene. There is, of course, a sensor operation and maintenance purpose for this in support of calibration and anomaly resolution. Since waveform clips are identified by a pulse number and a ray index, iray, finding a waveform clip through point data requires that the point data be resolvable to a ray index. This CMMD supports this via defining PF.Point.IrayNum.*TileNum*(ipt, j).

This CMMD permits reducing data volume in lidar enterprise level-2 data sets that have only a few waveform clips. Rather than defining all of the rays for the sensor, as one must for level-1 data sets, only those rays which have corresponding waveform clips provided get defined. Clearly, it is only in enterprise level-2 data that the possibility of using a point to find a ray and the associated waveform clips of interest exists. Only full fast time vectors exist in the wide band data for enterprise level-1 and from enterprise level-3 and up, there are no waveform

clips in the wideband data. When rays are defined for all of the elements on all of the arrays, it is done via SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray), where the names suggest that the ray indicator is globally unique throughout the data set. If only a sparse selection of waveforms is present, the data volume can be reduced by locally (that is specific to each tile) determined ray definitions, provided in SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). Both of these work by associating a value of array index and element index with a single value of iray in a pair of tables. The index, iray, is the same in both the array and the element tables. The tables are short for the tile specific case and relatively long for the global definition case. When there is a single array with only a small number of elements, the global approach is all that is ever needed. When the tile specific data iray definitions are provided, they overide the global definition table, if present, only for waveform clips. The points must use the global ray definitions whenever point ray associations are populated.

10.1.4.2 Scans

Lidar (for VRP) is a device and a process, which combined use the measurements of roundtrip photon travel time[49] and received power to probe a remote scene. The relationship between the transmitted and received signals yields information about the remote scene and intervening propagation medium. lidar instruments can be used in a number of modes depending on both the physical design and choice of operating parameters of the individual instrument. While the internal processing chain and detection mechanisms are key elements of such mode definitions and strong determiners of the accuracy and kind of information that can be derived from lidar data, the time scales of lidar operation also provide a framework for defining lidar collection modes and organizing the collected data. As discussed above in the timing model, the shortest time scales, those internal to the interpulse periods (IPPs) are devoted to measuring data in fast time that can ultimately be converted to range information. The next shortest scale is that associated with pulse to pulse time tracking, called slow time. Scene changes and viewing geometry may vary in this slow time scale. Multiple interrogations of the remote scene happen in this pulse to pulse time scale over ranges of one to tens of pulses interrogating a single spatial region. In modes where the lidar is set to dwell indefinitely on a single region, this slow time can extend to many tens of thousands of pulses. However, in a class of modes that is of interest to GEOINT data collection, it is necessary to sweep the lidar's field of view over an extended spatial region on the scale of several hundred to several thousand IPPs, somewhat dependent on the chosen PRF. These localized articulations of the beam direction are generally repeated periodic patterns. Multiple pulses can be aggregated over a single period of beam articulation

[49] Laser-based instruments that determine range from non-time-of-flight means are usually called laser scanners, not lidars. lidar and LADAR terms refer to the RADAR ranging through a time measurement principle, including very sensitive phase-based ranging.

that is called a "scan". During the interval of a single scan there is 'little or no' to 'only a small amount of' platform position change, but the region collected is much greater than the single pulse field of view of the sensor. Section 10.1.2.4 dealt in detail with the parameterization of the scan FOV variations. In this section we deal with counting and tracking of the scan cycles.

As explained in 10.1.2.4, the single parameterization of scans as elliptical cones of the beam direction surface with a scan phase as an independent parameter that can be varied for each pulse permits the creation of FOV movement patterns that range from arcs to linear traces at any angle to the velocity vector of the platform. The scan phase runs over an interval of 0 to 2π and can be used to guide the beam over a full cycle, bringing the beam ultimately back to the orientation relative to the sensor frame that it started in. For a scan pattern that traces out a full cone, this full cycle of scan is a natural counting increment. However, if the cone is flattened to a plane by making its minor axis zero, the motion of a full cycle consists of two sections. One moves the beam from an initial direction to a final direction such that the optical axis remains in a single plane and would trace a line on a flat target plane. The second section of the scan cycle reverses the FOV movement and may, in many cases, be considered to be naturally counted as a separate movement of the beam. Thus, in some cases, counting scans means counting two scans per cycle or 2π range of the scan phase. Since the phase can be assigned on a pulse by pulse basis, it is possible that the phase always remains between 0 and π. This represents linear scanning in only one direction with "carriage returns" performed between consecutive pulses "bracketing" phase just about π and phase just about 0. Each of these half cycles can also naturally be considered to be a single scan. So, it is evident that while the parameterization of scans covers a wide and useful range of beam articulations, the tracking and assigning of a count of executed scans does not mean always just keeping track of how many times the phase wraps around at 2π. Further, the parameterization permits the resetting of the scan cone axis (and the plane of the major scan axis, though this is not expected to be a common practice) on any pulse. Whenever the basic elliptical cone parameters of a scan are reset, that must be interpreted as grounds to increment the scan count.

With the arbitrariness of what is considered "natural" scan counting and the necessity of changing scan counts when the scan parameters change, we are led to ask, "When does counting scans matter?" The lidar model answer to that is that scan tracking matters when groups of pulses that have been deemed to be part of a single scan are, as a group, treated in a way affecting a block contribution to a final product. For mapping related lidar applications, this means that when the points derived from the ray-range determinations of a scan are given a common adjustment, we need to be able to book keep that adjustment with the measurements derived from that scan. [We are referring to supporting scan to scan registration.] It is interesting to note that this view makes whatever criteria went into counting off the number of pulses attributed to a single scan irrelevant. Consistency of what product content proceeds from the scan and what we have documented about the processing or derived positional estimates in that same content is all that matters. The processor can determine what degree of spatio-temporal adjacency deserves to make measurements part of a single scan and what degree should be treated as consecutive and

separate scans.[50] All the data definitions have to do is support documenting the consistent tracking of geometric or radiometric adjustments at whatever scan blocking level was decided in the processor. This tracking is performed via the scan index[51] "is" that appears in certain scan related parameters and should also be used in tracking the impact of local registrations and calibrations applied to the lidar data. Once such corrections have been rolled up into higher level aggregations, the scan level data are of historical use only, and could be eliminated from the data set without impact to downstream processing and exploitation. Because there may be several processing steps that occur before this state has been reached, and because the assignment of pulse sequences to scans can be retained at little data volume cost affording savings to downstream processes that would otherwise have to re-compute the assignment of pulses to scans, the data definitions of the lidar conceptual model permit passing the pulse to scan map within data files. This is done via SC.Pulse.Scan.Num.*TileNum*(ip, j). Scan level geometric uncertainty data includes error covariance support data for position and line of sight as well as nominal nadir stats. Scan level geopositioning adjustments are given via PF.Adjustmts.Mean.Scan(is, j) and PF.Registn.ScanDat(2). Lidar points can be traced to the scans in which they were collected via the pulse number so that point groups can be collectively treated as "scans" of data.

[50] But this should be limited to the pi vs 2 pi cycle question for completion of a scan articulation of the beam. Scans are always composed of strings of consecutive pulses. We have only got control over the starts and the stops, that is, where we break the IPP sequence of a collection operation to mark individual scans.

[51] The scan index need only be a unique number assigned to a range of pulses that make up a single scan (i.e. a π or 2π range of scan phase in the scan parameterization model). However, it is highly recommended that population of the pulse to scan look up table in SC.Pulse.Scan.Num.*TileNum*(ip, j) make the scan index be a sequentially incrementing unit step integer series with starting value "is"=1 at pulse 0 of the collection and incrementing each time the last pulse of a scan has been collected. The last pulse of a scan is recognized by the following conditions:

1. If the scan cone parameterization of the next pulse is different, or
2. If the scan cone parameterization of the next pulse is the same and
    a. The pulse has non- zero scan phase and is followed by a pulse that has zero scan phase
    b. The pulse has non- zero scan phase and is followed by a pulse that has a lower value of scan phase. This permits that a scan could be started "mid scan" in terms of phase without confusing the counting process, but phase can only increase during a scan.
3. The pulse is the last pulse of a collect

The scan index will then indicate the temporal sequence of the scans. Since there will be many pulses per scan, the number of scans will be far less than the number of pulses.

10.1.4.3 Subswaths

For very high frequency pulsing of the illuminator (thousands of Hz), moderately high scan frequencies may be achieved (tens to hundreds of Hz). Groups of scans may be collected into "subswaths". There are two ways this may occur.

One way for collecting scans into a subswath is a natural consequence of the commanded lay down pattern of scans. This is the "sensor based" subswath grouping and an example might be the case where the scan pattern is a group of nominally straight scans parallel to the track of the platform. Only a few such scans can have most of the projected ground coverage be contiguous pieces of ground compositable with adjacent scan components of the subswath and having data common to a single stretch of platform ground track. The next scan would have no common projected coverage with the first scan of the previous group and so could logically be said to be the first scan of the next group of scans comprising a "subswath." See figure 10.1.4-4 for a motivation of natural along track subswath scan aggregations. In the diagram section "c", a scan is labeled with an "X" to indicate that it can't be a continuation of the subswath since it has no along track overlap with all of the other scans in the subswath. The scans that might compose a single "collection" or "sensor based" subswath are identified to illustrate the concept.

The other way subswath aggregation of scans can be done is processor based. An arbitrary integer number (we refer to this as the "scan rule") of scans might be collected into a single sub swath so that parallel processing can divide the totality of collected scans into groups whose size is limited by memory management considerations. Another criterion that may be adopted in deciding how many consecutive scans to treat as a subswath is the need for adaptive registration. If it is known a'priori that a single adaptive registration can lead to required geometric fidelity of a product that has been blocked in groups of some number of scans, that level of collection of scans could be designated as a subswath. The number of scans to be put into a subswath on this basis would be derived from PRF and correlation model metadata.

Subswaths that are processor defined could be introduced as intermediate data aggregations in support of parallelization to achieve a processing latency reduction. To support this strategy option, subswath designations are 2 dimensional in the lidar data model. Suppose we define a first layer of processing that creates a lot of small sub swaths. Then the index of the subswath layer, "iswl", is equal to 1 and possibly there are, say, 128 subswaths. These may have registration based adjustments determined and the 127[52] adjustments would be stored for subswath pairs from the subswath layer designation of 1. In the next iteration of processing, these 128 subswaths might be combined to form a reduced number, say, 32 subswaths with a layer index of 2. These derivative subswaths would have (32-1) different adjustments and none

[52] (the number of pairs of adjacent subswaths)

may be the same as in layer 1. A third iteration could result in 8 subswaths with a layer index of 3 and a final combination would be done to create a single final output swath. See the illustration in the PF.Overlap.SubSwath.ID1(j) detailed definition paragraph for a picture of how subswath levels or layers might be used to rollup results from multiple processing nodes to the final swath. If these subswath registration processes communicate via a lidar file, we need to be able to distinguish separate subswaths by level. This is especially important when the adjustments are only determined and not applied in the intermediate steps. This avoids the situation where the final set of subswaths might start at number 160 and count up to 168. Since the ranges of indices for arrays are separate metadata items in this lidar conceptual model, it is possible that implementations of a 3-D array can be made effectively 2-D by defining "iswl" to only have a range of 1 to 1 when appropriate.

Figures 10.1.4-3 and 10.1.4-4 illustrate scans grouped into subswaths. The cross track scan subswaths contain the same amount of data in time and have common levels of decorrelation in the platform position data. These are the second type of subswath groupings. The along track scan subswaths are grouped by commonality of along track ground coverage. These are of the first type of subswath groupings. An "x" marks the next along track scan that is determined to be "one too many" and thus is replaced by the start of the next subswath.

Subswaths once formed, can be adaptively registered (determination/application of block adjustments) provided that there is overlap in the coverage of adjacent subswaths. This level of registration conditions the data for the final aggregation into a swath, which is the level of data aggregation associated with a single lidar collection operation. A swath is normally associated with the position history of the sensor platform over a collection operation, defined as a single uninterrupted period of data collection and a single setting of the sensor's collection parameters (such as scan control, PRF[53] control or ambiguity number, focal plane sensitivity, range timing resolution,…).

[53] Per section 10.1.1, the PRF may be variable during scanning.

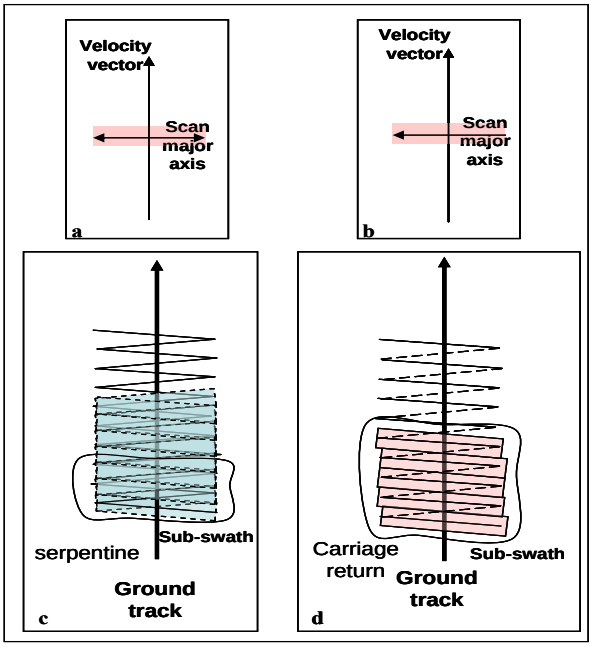


Figure 10.1.4- 3 Example Cross-track scans in subswaths. a) sensor frame serpentine, b) sensor frame carriage return (half cycle), c) earth frame serpentine, d) earth frame carriage return

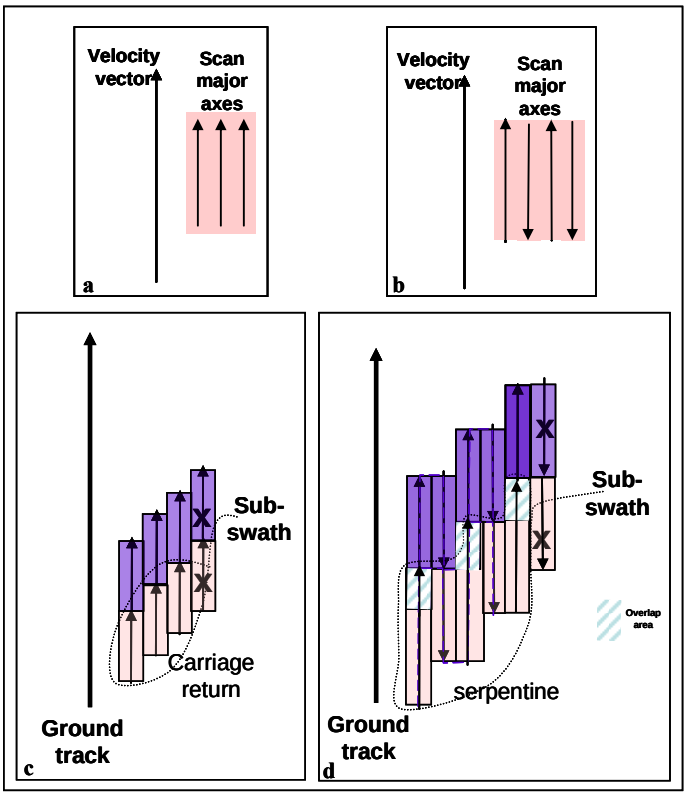


Figure 10.1.4- 4 Example along-track scans in collection subswaths. a) sensor frame carriage return, b) sensor frame serpentine, c) earth frame carriage return, d) earth frame serpentine

Documentation of collection subswath parameters in this conceptual model includes the number of scans per subswath at each level, PF.Subswath.NumScans(iswl), the list of scans in each subswath, PF.Subswath.ScansList.*iswl*(isw,it,j), and the number of *collection level* subswaths per data set, SC.Subswath.TotNum. [At enterprise level-1, there is one data set per swath, at higher levels, this is selectable.] The number of subswath layers is given by PF.Subswath.Nlayers. The number of subswaths in each layer is given by PF.Subswath.NumSubswaths(iswl). SC.Subswath.NumScansPer gives the number of scans per collection subswath.

The strategy for using and documenting the 'processing subswaths' is as follows:

- Use as many subswaths and subswath layers in the processor as needed to achieve the higher priority between processing time latency reduction and internal registration to improve product quality by reducing geometric distortions due to uncorrelated errors in assigning XYZ values to points in adjacent regions of the collected scene.
- When the strategy of latency reduction includes delaying the application of internal registration corrections to the exploitation phase on demand rather than building them into the output points, the corrections need to be stored on the basis of the regions for which the corrections were derived.
- When the ULEM parameters for error covariance and decorrelation are revised to incorporate the effects of having performed internal registrations, the new error propagation data values will be derived from the data blocking scheme and the size of corrections/residuals determined from performing the internal registrations.
- It is intended that data only be saved to a CMMD compliant data set at a stage where the processing of subswath levels has been completed. Thus, tiling (see section 10.2.1) will not be applied to the artificial processing subswath boundaries and the point duplication due to assigning single lidar points (or synthetic points) to both adjacent processing subswaths of a subswath pair will never be an issue.
- We can easily require that scans in a subswath be spatially adjacent. For uniform progressive ground scanning, spatially contiguous scan blocks will also be sequential. So, in that case it is trivial to have the scan index range within a subswath be consecutive scan index values of "is". But when ground regions are revisited, as in modes designed to collect nadir diversity, spatial adjacency does not imply temporal adjacency. So temporally labeling **scans does not lead to** consecutive scan index values within subswaths in all cases. However, blocks of scans within any spatially contiguous subswath will be consecutive temporally, so we can completely specify the content of a spatial subswath with a *list* of several ranges of first and last scan index values.

10.1.4.4 Overlaps in lidar Data

Overlaps in lidar data collection and processing can be used at all sub-levels of aggregation of level-3 or lower products. There are three reasons for having spatial zones of duplicate *collection* coverage.

- The first is to provide margin against the possibility that data voids may result from unexpected deviations from platform position and attitude, or errors in knowledge of sensor attitude and position during collection.
- The second is to support registration of limited FOV portions of a lidar collection into an overall product frame so that the impact of unknown errors, including refractive variations of the atmosphere, can be minimized.
- Lastly, overlap is a method of adding geometric diversity to single collection operations. Such diversity improves the probability of detecting objects by overcoming obscuration and statistical limitations.

There is also justification for a processor to work on data blocks that overlap spatially, even if the data block does not contain collection overlaps. The primary reason for this would be to reduce the propagation of relative positional biases within a large point cloud due to the various component blocks having internal location errors which are well correlated, but the blocks have mutually de-correlated errors between them. This is the natural consequence of collecting a large data set over time and having local data collected within a decorrelation time scale but the entire data set spanning multiple decorrelation time intervals. Massively parallel processing architectures will naturally take up data regions of both correlation levels as the point cloud is built up from individual pulse returns.

Figure 10.1.4.-5 summarizes the ways in which overlaps can arise and the purpose for creating these.

Level of Overlap	Purpose: Diversity	Purpose: registration	Purpose: void suppression
Sub-swath to Sub-swath overlap between collection sub-swaths Requires at least partial scan- to scan overlap of at least one scan in adjacent sub-swaths		X	X
Sub-swath to sub-swath overlap between processor subswaths Sub-swath to sub-swath registration could be supported merely by having a single scan counted as part of each of two adjacent sub-swaths. Works for cross track scans and processor defined subswaths.		X	
Scan to scan overlap within a collection sub-swath	X	X	X
Pulse to pulse overlap within a scan •Supports integration of array FOVs to scan FOV		X	X

Figure 10.1.4- 5 Scan overlap levels and possible purposes

The diversity cell for pulse to pulse overlap has been left blank and the registration cell has been marked. Some might object to this judgment as pulse to pulse overlap is definitely a method of improving the statistics of limited regions in the scene through multiple interrogations, a process we have described as a diversity benefit. Further, over short times between groups of pulses the position and attitude of the sensor is nearly constant, so that the risk of mis-mapping data without performing a registration is low. Further, unless the focal plane array is large enough, there may not be enough data in a single array FOV to support registration to an accuracy that is improved over the assumption of a steady platform. Clearly, the design parameters and performance data need to be assessed on a data set by data set case to determine what the necessary processing strategy is and what the collection spatial redundancy needs to be to meet coverage rate objectives and geolocation accuracy objectives. That means that the population of those cells in figure 10.1.4.-5 can only be representative and that a unique such table could be generated for every processed lidar collection. The motivation for the current cell marking is that intra-scan pulse to pulse diversity is minimal, leading to little change of viewing angle. Also, if consecutive array FOVs are placed into the product without adaptive registration, accuracy is necessarily sacrificed, especially when scans incorporate significant off nadir sampling where the shifts in the scene sampling regions are exaggerated by projection effects. While the sacrifice may be small compared to the investment to find the correction, the box is marked as a best practice from a quality point of view.

The next point to be discussed is why there should be overlap descriptions in the metadata. The early LAS formats completely ignore overlap information. However, this fact cannot be

taken to suggest that an assessment of overlap information utility was performed and that such information is of no value. Populating overlap information in lidar files can be a significant convenience, particularly if multiple applications are performing the early stages of data processing. If the collection parameterization is designed to support overlap at any of the levels where there is opportunity and there is no way of representing this to the processing applications, the locations of such overlap would have to be discovered via an intensive analysis of the wideband data. Furthermore, once done, if there is no way of passing this discovery to subsequent modules that may have use for overlap information, it would have to be performed again. Thus, supporting overlap documentation in the lidar model is not only a convenience, but part of a strategy of reducing processing latency for large volumes of collected data. In addition, overlap and registration go hand in hand. A major effort[54] in processing lidar is the registration of its parts. When the processor has made its own overlaps, the knowledge of these overlaps is necessary for a rigorous statistical assessment of geolocation errors. Judgments as to whether reprocessing could lead to worthwhile geometric accuracy improvements can only be made if the overlap information and use history is available. Lastly, overlap data may or may not be thinned out of level-3 products and understanding point density non-uniformities is enhanced when the overlap contribution is available to exploitation tools.

Factors that affect the required parameterization of overlap were incorporated into the preceding discussion. These include nominal amounts of overlap, the levels at which the overlap occurs, the nominal locations of the overlap and what utilization was made of the overlap.

- For registering subswaths the following PF.Overlap.xxx items constitute overlap description metadata:
  SubSwath.IDL1(j), Subswath.IDL2(j), SubSwath.ID1(j), SubSwath.ID2(j), and SubSwath.Dat(j).
- For registering scans the following SC.Overlap.xxx items constitute overlap description metadata:
  Scan.ID1(j), Scan.ID2(j), Scan.X_Lo(j), Scan.X_Hi(j), Scan.Y_Lo(j), Scan.Y_Lo(j), Scan.Dat(j).
- For registering pulse FOVs within scans the following SC.Overlap.xxx items constitute overlap description metadata:
  Pulse.X_Lo(j,k), Pulse.X_Hi(j,k), Pulse.Y_Lo(j,k), Pulse.Y_Hi(j,k), PulsePrcnt.
  The j,k indices here are over the pulse pairs and the scans. This data design exploits the fact that only the nominal pulse to pulse overlap is needed to support pulse registration in the formation of scans, even if the implemented registration involves pulse steps of greater than one. The overlap coverage of non-consecutive frames is derivable from this data.

[54] Applies to large area VRP collection that motivated the creation of this document.

Note that we do not need an additional list of overlapping scan pairs for each subswath pair over and above SC.Overlap.Scan.ID1(j) and SC.Overlap.Scan.ID2(j) . The trick is that there is scan to scan registration used to assemble each subswath, so all scans involved in that process are for a single subswath [specified by a value of the pair, (iswl, isw)]. There is also scan to scan overlapping used to do subswath to subswath registration. But for the subswath to subswath registration, the scan pairs have different isw match ups, and the processor may do the pair registrations in a lump, rather than pair by pair. So we have SC.Overlap.SubswathPrcnt for inter-subswath scan to scan registration, SC.Overlap.ScanPrcnt for subswath internal scan-to scan options, but only the one SC.Overlap.ScanID*n* list set.

10.1.4.5 Tracing Corrections to Point Geolocation Derived by Processing in lidar Metadata

In sections 10.1.4.2 to 10.1.4.4 we have described how pulse level data is collected into scans, scans are collected into subswaths and subswaths are collected into a swath level product. We have described how overlaps for the components of these hierarchical data aggregations are documented and suggested that registrations using these overlaps may be performed by the processor in assembling these hierarchical data collections within a lidar point cloud product. These registrations produce corrections that may or may not be applied at the point level. While it may be necessary to apply registration derived location corrections at the scan level (and possibly lower) in order to assure satisfactory 3-D “image quality”, the number of points in the larger subswaths can be immense and it may make more sense from the point of view of processing latency to not apply registration results to individual points until actually needed in an exploitation scenario. Whether or not the corrections have been applied at the scan, subswath or swath creation level, the PED may need to be able to recover the corrections at late stages and this CMMD provides data definitions in support of a robust capability with respect to inherent geolocation error patterns in lidar data.

Figure 10.1.4-6 illustrates the correspondence between processor functions and metadata items that support the documentation of registration based corrections used to build the swath level point cloud. This diagram shows how we can support registrations

- used to composite pulse frames to scans
- used to composite scans to subswaths
- used to composite subswaths to larger subswaths, and
- used to composite subswaths to a final swath product

This is a bookkeeping strategy that supports the documentation of registrations at each level where they performed. Each level is logically independent so that scans can be created

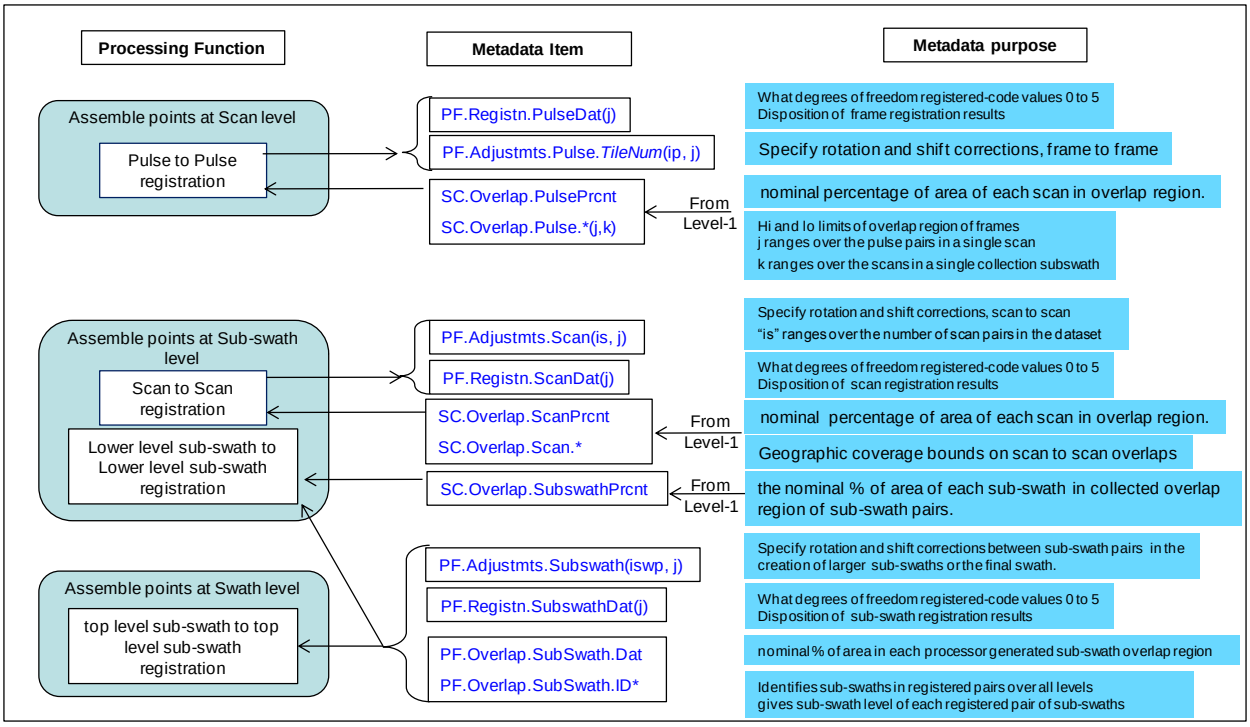


Figure 10.1.4- 6 Trace of adjustments derived from registrations used to compose scans, subswaths and the final swath.

without having to register pulse-frames and scans can be registered together into a final swath if that is what the processing solution requires. The advent of processor generated subswaths may come at great delay relative to the advent of routine scan to scan registration without being constrained by the CMMD but when the more extensive procedures are finally implemented, the CMMD is ready to provide the metadata "slots" to permit optimization of the latency and utility of point cloud products.

In addition to providing the bulk adjustments at each level of compositing points, the status of whether the corrections were already incorporated or are waiting to be used for maximization of geometric fidelity is provided. Note that the corrections are cumulative. This means that the lower level corrections must be applied before the higher level corrections are applied. Thus several different adjustments to individual scans must be applied to the scans individually, and then a single adjustment to the subswath that those scans collectively constitute would be made. In the case that there are multiple layers of subswath definition as indicated in the diagram within PF.Overlap.SubSwath.ID1(j), then the adjustments for the layers illustrated as more to the right side of the diagram would be applied after those due to registrations of subswaths in layers to the left.

Note that if any of these corrections are saved for post processing application, then the correct interaction with the Cartesian frame definition data at the tile level must be implemented.

To support being able to do this, the CMMD permits the adjustments at the level of forming scans, forming subswaths and forming the final swath to be only expressed in either the ECEF frame, or the local Cartesian frame of the tile in which a point, or points of interest are stored. There is a flag in PF.Registn.*Dat(3) to indicate which is the case (a separate value is given for Pulse, Scan and Subswath registrations. There is no burden for users when all of the corrections are applied in creating a data set. The tile level Cartesian frame data is simply used as if all the points were error free.

The case of having some corrections to apply before full geometric accuracy is afforded is a bit different. All points are actually stored as integer "index" data and are converted to metric positions via a linear transformation amounting to a universal scaling and a tile specific offset. After this transformation, the points are referenced to the Cartesian[14] frame of the tile. It is those "reconstituted" positions that are subject to correction via the PF.adjustments.* data items. We can expect that scan to scan and pulse to pulse registration adjustments will be applied for quality purposes so the correction data would be informational and likely little referenced. The subswath corrections will likely be not applied until specific points are selected. In that case, rather than needing to apply the collection to a large number of points, only a few points involved in a precision geolocation or mensuration exploitation scenario would need to have the subswath adjustments applied.

Given a point, the trace to a subswath or scan group to select the correct adjustment is enabled through population of the pulse number, ES.Point.PulseNum.*TileNum*(ipt, j). The scan that the point belongs to is determined via SC.Pulse.Scan.Num.*TileNum*(ip,j). PF.Subswath.ScansList.*iswl*(isw,it,j) identifies which scans are included in a particular subswath as specified by a subswath layer index (iswl) and a subswath index for that layer (isw).

10.1.5 Data Support for Geolocation error Propagation

10.1.5.1 Overview of Universal lidar Error Model (ULEM)

The NGA Sensor Geopositioning Center (SGC) has been developing a point cloud oriented model for rigorous positioning error propagation for lidar data. It is based on traditional photogrammetric arguments that amount to establishing some set of measurement uncertainties in the sensor frame and determining the sensitivity of earth frame locations upon these sources of error. An appropriate coordinate transformation is done in some software application and the result is an earth frame error ellipsoid for absolute (single point geolocation) or relative (two point separation) measurements as needed in a given mensuration task. The basis of the model is the assumption that typical products are collected over an extended interval of time and may have a range of errors that vary through the product so that a model, rather than a single error ellipsoid is needed to fully characterize geometric location error.

The SGC has taken an approach to the lidar error model that is sensitive to the metadata burden. The approach also accounts for the fact that we wish to be able to obtain extremely precise (in the sense that the position error is desired to be a very small fraction of collected scene extent) geometric measurements from point cloud products, but need much less precision in the knowledge of the error bounds. That is, getting position errors to the meter level does not require 3 digits of precision on the size of the error ellipsoid. In addition, the SGC accounts for things at a relatively high level of consideration, delegating many details of how to combine a plethora of potential hardware sources, most of which may be unique to the specific instrument design and operational details, to the vendor of the data to be rolled up into high level abstract errors. As a result, a relatively straightforward set of inputs to the model has been settled upon:

- Errors (7 variances plus associated correlations)
    - Sensor position variance (X,Y,Z)
    - Sensor orientation variance (2 angles for line of sight, one angle of orientation about the line of sight)
    - Variance in the range from sensor to measured point
    - Spatial correlation of these different errors, pair by pair
    - Temporal autocorrelation of any single measured source of error
- Collection geometry
    - Sensor position at the time of measurement (may be approximate)
    - Position of measured point whose uncertainty is to be determined.

While the formulations of the needed input data to such a process have not been well established yet, the functional definitions of the input data have been established to the point where they can be defined as metadata in terms of what they are supposed to represent and how they should be used.

- Σ = sensor parameter uncertainties matrix
- A = Partials of ground coordinates wrt sensor parameters
- **Single Point:**

$$\Sigma_{P1} = A\Sigma A^T = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ & \sigma_y^2 & \sigma_{yz} \\ Sym & & \sigma_z^2 \end{bmatrix}$$

- **Multiple Points:**
- The desire is to develop a full covariance matrix for the points

$$\Sigma_{P1,P2} = \begin{bmatrix} \Sigma_{P1} & \Sigma_{P12} \\ \Sigma_{P12} & \Sigma_{P2} \end{bmatrix} = A\Sigma A^T$$

- To populate the sensor uncertainties (Σ) for multiple points, the decorrelation as a function of time must be considered (ρ)

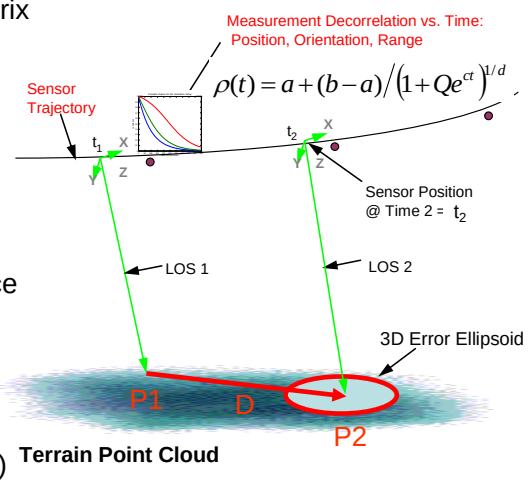


Figure 10.1.5- 1 Error model outputs for single and multiple point error propagation

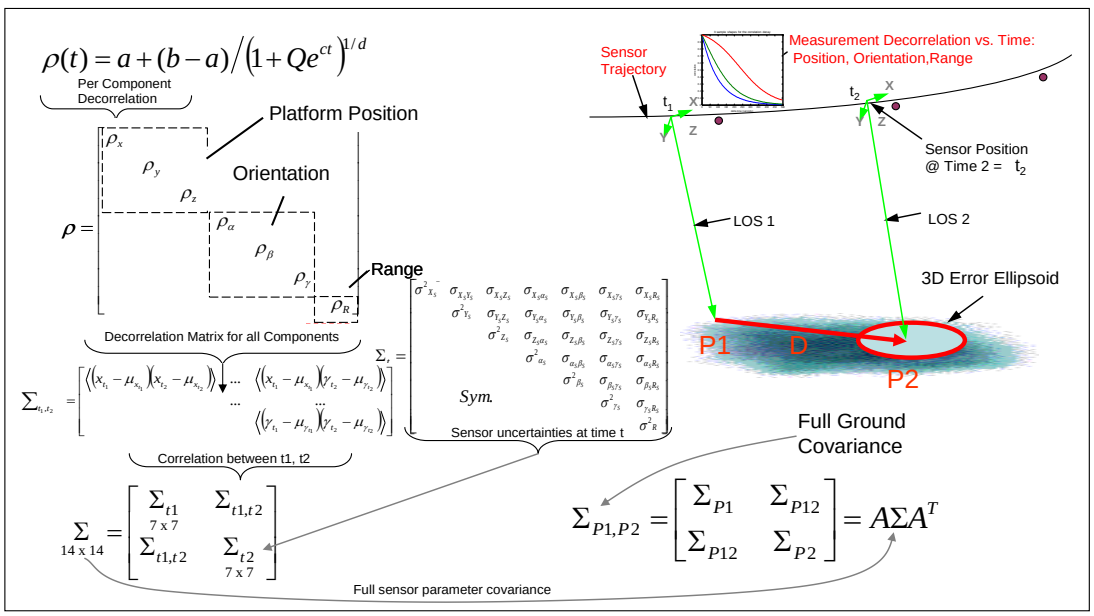


Figure 10.1.5- 2 A 7 x 7 sensor error covariance matrix is the basis for rigorous lidar error propagation. The time evolution of the auto and cross correlations in this matrix is needed to determine uncertainties in distances between points measured at different times. The CMMD supports a four parameter model of correlation decay[55] (aka decorrelation) for each element of the covariance matrix. The two point final covariance is $\Sigma_{Relative} = \Sigma_{P1} + \Sigma_{P2} - 2\Sigma_{P1P2}$.

[55] It is expected that the impact of internal registrations in the assembling/processing of point clouds will revise the decorrelation model parameter values from the values native to the trajectory and attitude filter outputs.

Note that this CMMD uses data to represent sampled values of the covariance matrix element components so that the covariance matrix for a time interval can be constructed for a point position in a point cloud. However, we use a parameterized *model* for the temporal correlation function belonging to the terms in the covariance matrix other than range cross terms. It is to be understood that the parameter values populated into a dataset will be the best *fit* values that tailor the CMMD's generic temporal correlation function model to the vendor's possibly more exact statistical data or model. The CMMD model will be a close fit to realizable temporal correlation functions, but it need not be exact to serve the purpose of supporting relative error uncertainty estimations for point to point mensuration procedures.

10.1.5.2 Conceptual Model Mapping and Allocation of ULEM Metadata

The Enterprise level-1 lidar data is always in the time domain and is preliminary to any processing to arrive at a point cloud. Enterprise level-2 and 3 products may contain point clouds that have been subject to a variety of degrees of processing and composition. The least processed point products or interim stages between products are the "raw" lidar points, which are the direct result of round-trip time to range conversion and transformation to a Cartesian frame. For these points, the time of *collection* is cleanly determined by the pulse on which the point was measured [in some cases there may be some ambiguity about the *roundtrip* time when a range ambiguous PRF is used]. Regardless of the range ambiguity number, there is little uncertainity about *which* line of sight to associate with the outgoing and return rays for a measured point in such a "raw" point cloud. Of course, the line of sight variances do apply and there is some uncertainty about what exactly the given line of sight for a particular pulse is. For such "pre-processing" or "raw" points, the error covariance data is supplied by metadata tags that have "Scan" as a delimited substring. See for example, ES.Scan.SigmaX(j). While some sensor processing may, in fact, not output scans as "raw" points, the scan is a sufficiently low level of data aggregation that its name has been recruited to indicate a version of error covariance data that is directly derived from single collection sensor and platform information alone.

It is a rarity that "raw" points would be present in a product intended for user exploitation. Products with "raw" points are generally precursor products to ones that have improved the "quality" of the point clouds by performing filtering steps such as denoising, deconvolution, artifact removal and averaging. In such products, it is common that the clear association of a pulse on which a point was collected may be lost, since the lidar points are often replaced by synthetic lidar points resulting from points collected over a possibly significant time interval. To document such multiple times (and positional smearing as well) attributed to a point, the data items PF.Point.Syn* are made available. Another common processing step, generally necessary for compositing all of the pulse frames into a final swath level product is registration. In a registration step, overlapping but geometrically adjacent blocks of a point cloud are compared for structural content in the overlapping region. Positional adjustments are derived from the cross correlation in the overlapped zone of such adjacent blocks. When applied, the point cloud on the aggregated level has been freed from some of the distortion due to uncorrelated sensor

position and line of sight error. The 7 x7 error covariance data representing the processing reduced distortions is supplied via some of the ES.Swath.* parameters. See for example, ES.Swath.SigmaX(j). The ES.Swath.* error covariance data must be interpreted as representing all of the processing and registrations that have been done to get to the current data set, even if a later downstream product data set might eventually have another level of registrations performed. Thus, like the use of the name, "scan", for the "raw" point covariance, we have recruited "swath" to indicate that all processing and registration effects accumulated so far are rolled up into this version of the error covariance data.

It is important to note that both the scan and swath level error covariances may be supplied for a single product. This is of particular value when the registrations have been done to adaptively determine the needed shift and rotations to make the product as seamless as possible, but the transformations may not actually have been applied. The metadata scheme permits that such information can be stored only and not applied. Such transformations would be applied only on a point by point basis as the coordinates of specific points are queried by a user or using application. This permits an enormous reduction in production latency and is a very likely strategy for processing of large data sets. Depending on whether the transformations are actually applied, one or the other versions of the error covariance data are appropriate and populating both could be a significant convenience as well as providing additional analysis support data.

Updating Geolocation Related Data for a Point Cloud

Operational circumstances may present the problem that lidar data is processed as soon after collection as possible to meet a latency requirement for a point cloud that permits greater than the final system geolocation error and later information becomes available that permits improving the geolocation accuracy of the point cloud. In such a case, it could be desirable to enable updating of the geolocation and uncertainty data without having to recompute the entire point cloud from the raw data. This CMMD recognizes that once improved sensor position and attitude data is available and it has been used to update point positions, the original sensor geometry data is of no potential use in the PED chain. {There could be O&M uses, such as tracking the magnitude of sensor geometry update corrections, that are easily performed without carrying the lidar wideband content.} Corrections due to post collection processing of navigational system data, ground truth or multi-look resection based adjustments that lead to block adjustments are performed by updating the tile reference frame data, the pulse indexed sensor position and updating the ES.Scan.* covariance (uncertainty) data items and indicating the status by setting PF.Swath.OffsetStatus flag to indicate the nature of the update correction source. On the other hand, corrections due to point processing such as internal registrations of single collection data or application of surface finding techniques that adjust the ranging calculation may result in actual changes to the individual lidar points within a tile and the modified covariance data to support this would be placed in ES.Swath.* covariance items. Since the ES.Swath.* covariance data are the net uncertainty data for a product, these items need to be recomputed if a level-2 or higher product is generated with collection time data and later updated

sensor position and line of sight data are introduced without recomputation of the points in the point cloud. The CMMD metadata are designed to support CONOPS wherein an update message may be sent to a custodian of a product dataset (e.g. a library) and a metadata update based on that message alters the error propagation and geometrical adjustments content of a file without the need to recompute lidar points. This is done by

- being able to label corrections as having been determined and applied or determined and not applied
- being able to indicate prior update activites via PF.Recon.*

In addition to the documentation of tile Cartesian frame origins and axes, which if updated would effectively revise the coordinates of all the points in the affected tiles, this CMMD supports documentation of earth frame adjustments consistent with the ULEM "anchor point" concept. PF.Adjustments.Mean.* represent six degree of freedom (3 shifts components, 3 orientation components) adjustments that can be applied to point cloud blocks at the pulse, scan and subswath levels to make rigid adjustments to the affected point cloud point locations. If the processing has determined that adjustments within scan and subswath levels needs to be variable, a grid of shifts (in excess of the mean shifts) suitable for trilinear interpolation can be populated into PF.Adjustments.Diffs.*. Implementers are free to use any of these approaches as is appropriate to their processing chain so long as the values of the data are consistently populated. An example of consistency is assuring that values of adjustments that are marked as "to be applied in mensuration" have not already be incorporated into the tile coordinate frame metadata values so that implementting the adjustment amounts to a double(and thus wrong) correction. Another example would be populating non-zero mean shifts into the mean shift adjustment arrays and then making the values of the "Diffs" arrays incorporate total shift, again leading to double application of adjustment data.

10.1.5.3 Physical Ranging Error Model for Lidar

There are three basic range measuring schemes with lidar. In linear mode [characterized by a detector current proportional to the incident photon flux], there is peak finding in a return waveform for an unmodulated short pulse and there is pulse compression for returns for a modulated extended duration transmit waveform. Various methods are available to time sample the return waveforms. Detector arrays with clocked readout electronics are a common and straightforward methodology, as is using the streak camera principle to convert photons to electrons whose time of flight within the sensor is exploited to spatially separate returns with slightly different arrival times within the sensor. For the case of Geiger mode photon detection, a short transmitted pulse is also used and the time that a diode is triggered into a nonlinear avalanche is recorded.

All three of the above methods have the property that a time is associated with a photon return flux from a single range. Different reflecting target ranges will have different time of reception histories. While the Geiger mode return time is less subjective than the unmodulated short pulse linear mode return time determination, depending on the sensor's timing and photon response, the uncertainty in the received time is a common element in roundtrip time measurement and is a universal reduction of the measurement process.

Similarly, in order to form a round trip time measurement, the time of pulse transmission must be measured. The round trip time is thus a time interval delimited by a transmit time measurement and receive time measurement. The uncertainty in the round trip time deduced from this pair of measurements is a direct determiner of the ranging error for any single pulse level raw scatterer range estimation based on recorded (time domain) lidar data. Because of the physical distinctions in the underlying measurement (estimation) processes, we separately discuss the origin of the common summary parameter, the uncertainty in the roundtrip time. Also, to reduce confusion and exploit the hierarchical naming scheme employed by this CMMD, we give separate names to the linear mode (i.e. continuum process) parameters and the Geiger mode (i.e. statistical discrete process) parameters.

Note on correlation of range uncertainty and sensor position and orientation uncertainties:

The ranging and sensor position determination are performed by completely independent subsystems of the lidar. If the navigational system were perfect, we would know the exact sensor position at each IPP index value. We might be wrong about the time that the sensor is at that position, but not about the position when the flash is transmitted and received. So, clock rate induced ranging error does not introduce a sensor position error so long as pulse indexing is used as the system timekeeping scale. This CMMD, by promoting data management in IPP index space, supports decoupling all ranging error from platform data errors. Consequently, this CMMD assumes no correlation between the ranging and platform position and orientation errors and does not provide data to convey range error cross correlations .

As a further note, correlation of errors is a distinct phenomenon from constancy of the uncertainties. Variation of the statistical error distributions is handled in the CMMD by tracking of covariance "snapshots", and there need not be any variation in the error distributions. But error correlation refers to individual measurement pairs having a level of consistency where subtantially the same error is accruing to each measurement, regardless of the ensemble from which the errors are drawn.

Geiger Mode Round Trip Timing Error

The receiver in a Geiger mode lidar is set up so that it can respond to just one or a very few photons arriving within an individual detecting element's energy accumulation time, which is generally an interval very short compared to the clocking intervals for quantizing the time at which a nonlinear avalanche response occurs. A clock marks time intervals and no response is recorded until there is an avalanche. Each clock interval has a reference time (commonly the

instant at the temporal middle of the interval). A photon whose avalanche occurs at any time during a clock interval is given a time marker which is the reference time for the clock interval in which the avalanche occurs. In this mode, no finer time resolution of the reception event is possible. Usually the avalanche occurs due to a photon received during the time interval, but it is possible that the avalanche may not reach a recordable level until a short time after a photon arrives at the very end of a preceding clock interval. For well designed and properly biased detectors, this delay is negligibly short and we can treat the avalanches as occurring at any instant within a single clock period and the photon arrival as effectively simultaneous with the avalanche event. The result of this process is that the moment of photon arrival is recorded as a discrete time of finite precision that is uniquely determined by the clock count at the interval in which the photon is detected. Fortunately, clocking intervals of the order of fractions of a nanosecond are achievable so that order few centimeters or less light propagation distances may be associated with realizable clock time discrete steps.

While the process just described is able to very precisely measure when a photon arrives at the sensor, there is a problem in that we can not determine when the detected photon was emitted. All we can measure is a time for the pulse emission during which the detected photon was transmitted. Were it to be the case that the duration of the transmitted pulse was less than or equal to the time measurement resolution of the instrument, the round trip uncertainty would be due to two of these time of flight measurement clock intervals. However, it is quite possible that the transmitted pulse has non-zero photon flux over an interval of many such clock steps so that in effect attaching a single time to the pulse corresponding to, say, its peak would result in substantial uncertainty in measuring the transmit time of the *detected* photon. Since laser pulses have a rise time during which early photons are emitted and a tail, or perhaps even a secondary pulse peak during which photons are also emitted, a sensor that can respond to single photons will often respond to photons emitted earlier or later than the recorded pulse time for transmission.

A figure of merit representing round trip timing confidence for such a sensor is the duration of the time interval during which 95% of the photons in one pulse reflected at a single range from the sensor arrive at the sensor. The interval in which 95% of transmitted photons from a single pulse are received after reflecting from a single range is a statistic about the round trip time distribution for lidar returns. The total number of photon arrival times covers the complete ensemble of the measurable values for the round trip time. Consequently, the percentage of photons represented in an interval is the cumulative probability of the round trip time falling into that interval. Thus, 95% of photons gives the .95P of measured round trip times and we may refer to this time interval as representing both the time to receive 95% of photons in a single measurement that can accumulate returns or the .95P time interval for a measurement with a very large number of transmitted photons and only one received. So, this is a statistic which has meaning even when the return flux is only a few photons for a single pulse if we determine it over a large number of pulse measurements. With a large ensemble of pulses, the distribution of recorded round trip times will become evident and this figure of merit for the sensor can be measured. Such a figure of merit will be a function of the operating parameters of the lidar such

as temperature, laser pulse duration, power and detector voltage bias. These are necessarily fixed quantities for a single collection operation. (When we change them sufficiently to significantly modify the range timing figure of merit, we must start a new collection operation.)

When the transmitted pulse envelope is asymmetrical about its peak, the 95% receive interval can vary a bit as its start time is varied relative to the peak. Thus, we generally need to specifiy which 95% interval to report. One option is the maximum duration 95% interval that can be found for the transmitted waveform, which would conservatively bracket the round trip times when the received data leads to low density local point returns. However, the maximum .95P interval would likely start at one extreme of the photon transmission time on a tail of the transmit time distribution. Being the maximum interval, it must necessarily be influenced by the least likely photon transmission times and it is not based on the expected value of round trip time. A decision on how to report the .95P interval should be based on the fundamental error propagation principal that error combination procedures formally apply to adding variances (squared standard deviations, that is about the mean). So we populate uncertainty data with standard deviation values and support confidence interval documentation with a multiplicative factor to get to the .95P interval's upper and lower error bars referenced to the mean (i.e. expected value). We place the range independent (see next section) standard deviation duration interval parameter in ES.Swath.GM_SigmaT and associated .95P factors in ES.Swath.P95GM_Fac(j,k).

The associated range uncertainty for raw[56] lidar points is defined at the scan level with a possibly different value for each sensor error covariance snapshot. The (potentially variable) refraction corrected speed of light is used to convert the round trip timing uncertainty to a range uncertainty.

Atmospheric refraction must be accounted for when accuracy requirements are so stringent that ray path curvature must be taken into account to compute accurate earth frame lidar point positions. However, the geolocation uncertainties due to errors in modeled ray deflection in the atmosphere are range and depression angle dependent and also second order in magnitude. Consequently further discussion and definition of supporting metadata is deferred in this version of the lidar CMMD.

<u>(GM) Range Dependent and Range Independent Ranging Error</u>

The range from the lidar sensor to a lidar point is determined by multiplying the round trip time measured by the sensor by half the refractive index corrected speed of light for the ray path. As discussed in ES.Swath.GM_SigmaT, the round trip timing error has contributions from marking the transmit time, marking the receive time, and fast time clock rate error. Of these, the clockrate contributes an error that grows with range, while the measurement of the time of transmit and the time of receive are independent of the range. Furthermore, the path averaged

[56] A raw lidar point cloud consists of natural lidar points that have not been filtered, decimated or adjusted in any way.

refraction corrected speed of light can contribute an error that accumulates with range. Thus, for a lidar that operates over a wide range of sensor to reflector ranges, it is necessary to treat the range dependent and range independent range error souces separately so that the correct combination can be selected for particular points, or groups of points when computing net range uncertainties in practice.

The range dependent component of range variance is

$\left(\sigma_{R_{rd}}\right)^2 = R^2\left[\left(\frac{\sigma_{\delta t}}{\delta t}\right)^2 + \left(\frac{\sigma_{\bar{n}}}{\bar{n}}\right)^2\right]$ where $R^2 = \left(\frac{c}{2}\bar{n}\tau\right)^2$, τ is the round trip time, and $\bar{n}$ is the path averaged index of refraction. The (fast time sampling) clock keeps time by counting *m* steps of mean duration δt and σ indicates standard deviation as the uncertainty measure. The range independent component of range variance is

$\left(\sigma_{R_{ri}}\right)^2 = \frac{c^2}{4}\bar{n}^2(\delta t)^2\left[\left(\sigma_{m_{Rx}}\right)^2 + \left(\sigma_{mTx}\right)^2\right]$. Here, m_{Rx} indicates the clock step index for the photon receive time and m_{Tx} indicates the clock step index for the photon transmit time. We provide $\sigma_{R_{ri}}$ on a covariance snapshot basis (to track any variation in $\bar{n}$) in ES.Scan.GM_SigmaR(j). The last two factors in the expression for $\sigma_{R_{ri}}$ are represented in ES.Swath.SigmaT. The key elements for computing range dependent range uncertainties are derived from SC.Swath.SigmaClkStep, which permits computing $\sigma_{\delta t}$, and ES.Scan.SigmaNbar(j), which carries $\sigma_{\bar{n}}$.

For the data model to be broadly applicable, it is not appropriate to make a'priori assumptions about whether the range dependent or range independent error sources dominate. Whatever the case is for a product or a point within a product, that must be determined by the values of the metadata.

We are faced with options regarding how we treat the metadata definitions for the timing errors and the consequent ranging errors. We choose to support convenience and flexibility by permitting the population of the atmospheric parameters from which the path averaged speed of light are derived and not directly storing the result of that calculation, which is performed at the level of the time sampling associated with capturing "covariance snapshots". We then provide the range dependent round trip timing error and the combined transmit and receive time uncertainties through separate items SC.Swath.SigmaClkStep and ES.Swath.GM_SigmaT. Finally, only the range independent range uncertainty is directly populated into ES.Scan.GM_SigmaR(j) on a "covariance snapshot" basis. By dividing this by ES.Swath.GM_SigmaT, the range independent timing uncertainty, we obtain the path averaged speed of light corrected for refractive effects during the snapshot interval. This derived speed of light could be used to rescale the range dependent timing uncertainty to a range dependent range uncertainty.

Typically, the range independent range uncertainty prior to processing to reduce positioning error, ES.Scan.GM_SigmaR(j), will be somewhat stable for a collection operation. The largest contributor to variation would be due to variation in the nadir angle of rays during a collection operation. Even weather is unlikely to produce substantial fluctuation. When ES.Swath.GM_SigmaT is many times the time quantization error, the impact of the receive time quantization and the pulse profile (range spread function) are very different. We create point positions with the discrete pulse transmission time and photon arrival times recorded by the sensor. In XYZ space, these are simply geometric points. The raw points in a Geiger Mode lidar point cloud represent the positions where measurements succeed in having a result. Think of lidar point cloud formation as the sensor checking a voxel in scene space for its status as being occupied or empty. The receive time quantization errors determine the voxel size (in the range direction) for volumes of local position uncertainty, which we focus on here by ignoring, for the moment, the fact that the voxel itself could contain returns from range ambiguous[57]positions. The receive and transmit time interval reference "ticks" determine where the voxel is in scene space relative to the sensor. There is no guarantee that there is a scatterer physically present at the point position implied in the lidar measurement process. The receive time quantization errors give a lower bound to the range error of this measurement process.

There is another categorization of geopositioning error sources that is important in the error model. This categorization has two classes of error sources: 1) those that lead to a shifted position for points in the scene relative to a true absolute position and 2) those that lead to feature signature smearing as evidenced by spreading of the point return positions for a single motionless scatterer in the scene. Table 9, below, lists error sources for Geiger mode in such a breakout.

[57] This statement refers to the ability to resolve different positions in the face of labelling all returns in a volume with a single position, namely the center of the volume. Because multiple positions are given a single label, this appears to be a formal case of ambiguity, but we reserve the usage of the term "ambiguity" to range ambiguity, a phenomenon encountered when the ambiguity number is positive. Range ambiguity due to ambiguity numbers larger than 0 is a separate topic.

Table 9: Geolocation Error Sources by PSF and Non-PSF categories.
All entries but platform sensor position and attitude, and the ray path shape contribute to ranging error in Geiger Mode.

Geolocation Error Source Types
(GM)

Uncertainties that Lead to mislabeling a measurement position in the earth frame	Uncertainties that Lead to attributing a response from another position to the measurement position
Platform/sensor position	Transmit time quantization
Platform/sensor orientation	Receive time quantization
Speed of light on ray path	Pulse profile (fast time history)
Shape of ray path	Detector element field of view & Optical PSF
Clock rate	Cross talk
Internal delay calibration and trigger thresholds	These errors lead to a smearing (Point spread function) of multiple point responses to a single point scattering element. That is, the cluster of points due to a single scatterer is spread out in a point cloud.
These errors lead to a shift in a lidar point position (measurement position) from the error free case.	

The conclusion of these observations is that isolated points in a raw lidar point cloud each represent the center of volumes where a photon reflection is computed to have occurred. That is, any point in this volume could have been where the photon return reported as the point at the center of this volume actually occurred. Of course, this cannot correspond to an actual scene position for the photon reflection unless the path hypothesis is correct. We can build the individual photon detector element field of view into this uncertainty volume, but we need to further assume that the shape of the assumed ray path is correct, and this assumption can be wrong whenever there are multipath or refractive effects that are not included in the ray path model used to compute the point position. But even in the error free case, **the lidar point is where a measurement nominally takes place** and that need only be somewhere on a scattering surface. The uncertainty in the location of points in a lidar point cloud can be computed from data about the sensor and its position and attitude. Even considering points due exclusively to direct reflections from objects in the scene , the uncertainty of the point locations (the positions of measurements) is distinct from the uncertainty in the position of the objects being sampled at those points, because the points in the point cloud need not be at the position of a scatterer. The geometrical position, shape and orientation of that surface must be deduced from a large number

of points.[58] When the surface is not orthogonal to the "look ray", and especially if the surface normal vector is not constant over the cross range extent that had to be averaged to do surface estimation, there is going to be uncertainty in the surface determination. The more cross range averaging that is done to statistically improve the range estimate, the poorer the cross range resolution will be of the point cloud[59] that has been processed to reduce the sensor's range uncertainty impact on the raw lidar point positions. Even on a flat surface orthogonal to the look direction, this leads to reduced localization of intensity or reflectivity like estimates. But when the cross range averaging samples surface elements actually having different ranges, a low pass filtering of surface relief is also generated. This necessarily compromises the estimation of the location of scene range discontinuities (typically associated with roof lines and overhangs). Since it is possible to implement adaptive cross range averaging schemes for surface finding, it is difficult to provide cross range resolution information everywhere that is accurate without generating prohibitive amounts of data. The solution for this challenge adopted in this CMMD is to indicate the minimum and maximum over the product for the diameter of the effective cross range averaging circle. The effective cross range averaging circle has the same area as any rectangular region over which the averaging might have been performed. This will be found in MD.Qual.XrangeAvDia(j).

We can consider that if the pulse transmit time were error free, for each output point there would be an uncertainty volume that is approximately a disk whose diameter is due to the cross range field of view of the detector element and thickness due to the receive time quantization. All points within this volume, if they were the position responsible for reflecting the detected photon, would be given the location of the center of the disk. But some times the recorded pulse transmit time is earlier than the actual emission time of the detected photon. That makes the sensor determined round trip time an over estimate and we place the point at a longer range from the sensor than the true round trip time (if we knew it) would have indicated. And, sometimes the recorded pulse transmit time is after the emission time of the detected photon and the converse is true; the computed range is an under estimate and we place the point closer than the true scattering position for the detected photon. The consequence of this is that when we can time resolve the transmitted pulse over a number of fast time samples, the Geiger mode lidar

[58] Synthetic lidar points can be generated into a derived point cloud such that they represent a hypothesis of a local surface via a decimated pattern associated with a large number of the raw returns. The positional uncertainties of those synthetic points will be related to the "fuzziness" of the raw data as well as the certainty to which the hypothetical surface position can be identified in the fuzzy raw data. The algorithm that generates such synthetic points would need to provide the new post processing (i.e. "swath" level) uncertainties for the positions of the points it produces that purport to follow feature boundaries in the actual scene.

[59] Raw point clouds, by virtue of being points, are formally "super resolved" in the sense that the samples are nominally associated with geometric points and not regions of the size of a point spread function. However, nearby points in processed point clouds may be correlated and not be indepdendent samples of scene structure. The rendering of scene feature details by such points is not necessarily at the level of detail suggested by the separation of the point samples or the native resolution of the lidar's optics.

point cloud, even in the absence of noise, results in a fuzzy rendition of the scene's scattering surfaces. For single quench per pulse operation, each detector fires at most once per pulse. We place only a single point along the detector element's line of sight or we leave that ray without any points. We have no clue, on the basis of one pulse alone, when relative to the recorded pulse time the detected photon was actually emitted. Thus, there is no basis for correcting the range of a point for a single isolated PCD.

Processing can provide some mitigation of this problem, however. The fuzziness of the point cloud can be reduced in a processing step that attempts to guess the range for a locally flat, possibly oblique surface by estimating the range profile of the return pulse over a number of nearby and consistently selected PCDs. When the model for the pulse profile is accurate and the local density of the raw point cloud is high enough, a substantially correct scattering surface can be estimated. When the model is not right, either the fuzziness is sub-optimally reduced or the range to the scattering surface has a residual local bias, which can introduce artifacts of similar challenge as the original point cloud fuzziness. This can include erosion of edges, introducing unfaithful surface renditions and local scale geometric distortions. This means that we need a post processing figure of merit describing the impact of any surface finding algorithm in the presence of a native ES.Swath.GM_SigmaT to support mensuration of processed points. Such a figure of merit should depend on the goodness of fit of the range response model to the local point cloud. It should be scaled to a number between 0 and 1, indicating the fraction of the range uncertainty associated with round trip light travel and ES.Swath.GM_SigmaT that is residual after processing. Since the quality of local fit of the range response model is strongly dependent on the local point density and scene structure, we need to make the process of determining range uncertainty in the point cloud spatially variable in principle. Rather than storing values for this for every point or for thousands of voxels, we provide a way to map this to a local quantity such as raw point density, via a table. The parameter that serves this purpose is ES.Swath.GM_SigmaRfrac(i,j). In point clouds where the points have been processed to represent probable surface locations, ES.Swath.GM_SigmaRfrac(i,j) describes the typical dispersion of range about the physical location of the scattering surface. The best case residual range error is the range associated with the discrete fast time sampling quantization of the lidar. If the precision of the range calculations permit locating points at various ranges within the range cell (minimum voxel extent), a random distribution of points one voxel thick will still be seen in point cloud products. Whether or not this minimum is achieved, if the processed points have no range bias, the shell would be centered/centroided about the position of the actual surface. But we could not know that unless we had a'priori knowledge that the surface were smooth on spatial scales long compared to the variation in the shell boundary fit function. Sufficiently short radii of curvature and low sample point densities lead to a maximum uncertainty in the surface location estimate.

Unmodulated Short Pulse Linear Mode Round Trip Timing Error

{Deferred}

Pulse Compression Modulated Long Pulse Linear Mode Round Trip Timing Error

[Deferred]

10.2. Lidar Data Strategies

10.2.1 Lidar Physical Data Blocking: Tiles

10.2.1.1 Tiles: What and Why

The data at pulse, scan, subswath, and swath levels is *logically* connected within those data divisions. One of the challenges of lidar data is managing very large amounts of data. It can be collected at high rate in large collection operations and even after processing can be massive when data from a single collection operation is stored and disseminated in a single volume. When there is logical connection between wideband data groupings [natural for data connected by a single collection activity of a single lidar sensor], it is a major inconvenience to either have monolithic files with all the groupings together for which the IO is burdensome due to file size or to create a plethora of smaller files and push the burden of recognizing and managing these groups off to external software or library systems.[60] For the purpose of *physically* blocking the data for I/O management, it is convenient to collect the wideband data into "tiles". The narrow band (meta-)data is not a significant problem and with this data model is naturally separated into readily retrievable groups. A significant benefit of tiling is the possibility of reducing the bit volume for XYZ values. While XYZ data can be given arbitrarily high precision at great expense to data volume, the fact that there are errors in the XYZ position makes only the leading bits of such bloated XYZ values meaningful. The precision of the XYZ word need only support the number of (sparse) grid positions possible within a tile, as opposed to the full data set.

The approach to creating manageable wide band data is to observe the natural association of atomic data items and to consider how repeated associated groups of items can be sorted and stored as array like structures susceptible to convenient and fast I/O operations. Data that is likely to be used together should be stored together and subject to as few cache misses as possible. Once handy chunks of data can be given a single name, or can be retrieved with a minimum of data striding, all we need to do is provide a map of that data in the metadata, the data names for retrievable chunks and we have a tiled data set. An overview of the main features of the tiling construct used in this CMMD is shown in Figure 10.2.1-1.

The metadata are capable of describing sorting at the tile level and the native global sort before the first tiling (see PD.Sort.NativeOrder, PD.Sort.NativePri(j). This means that data are permitted to be sorted differently in consecutive tiles. While the bounds of data in a tile are described in the metadata (see PD.TileMap.Lat/Lon(j_tile,k)), the notion of sorting on a larger scale than a tile can only be derived by global analysis of the tile sorting and bounding descriptors. Nothing prohibits a software tool from re-tiling a file so that data in the new tiles

[60] Four arguments for tiling are that the appropriately I/O sized data groupings in tiles 1) avoid the unfunded enterprise mandate when external management of fragmented files is assumed, 2) improves processing and display system performance, 3) is consistent with other metadata blocking patterns, 4) Makes sorting problems "local", simplifying software and improving performance

may be derived from multiple tiles in the precursor data set. However, two constraints must be applied to any retiling process to assure traceability of the tile mapping.

- "Up-tiling" (combining smaller tiles to create a larger tile) is only permitted for temporally adjacent tiles (enterprise level-3 and below where temporal adjacency implies spatial adjacency). For level-4, spatial adjacency is the basic rule as temporal adjacency is often lost in the merging process.
- "Sub-tiling" is only permitted for tiles that have horizontally variable parameter (such as Z or intensity) sorts performed as 3rd or later "then by" sorts. So, in particular, "Z" and "intensity" sorts must be after sorts on time of collection and X or Y position.

This document gives the parameters that are used to describe the old and new tiles alike.

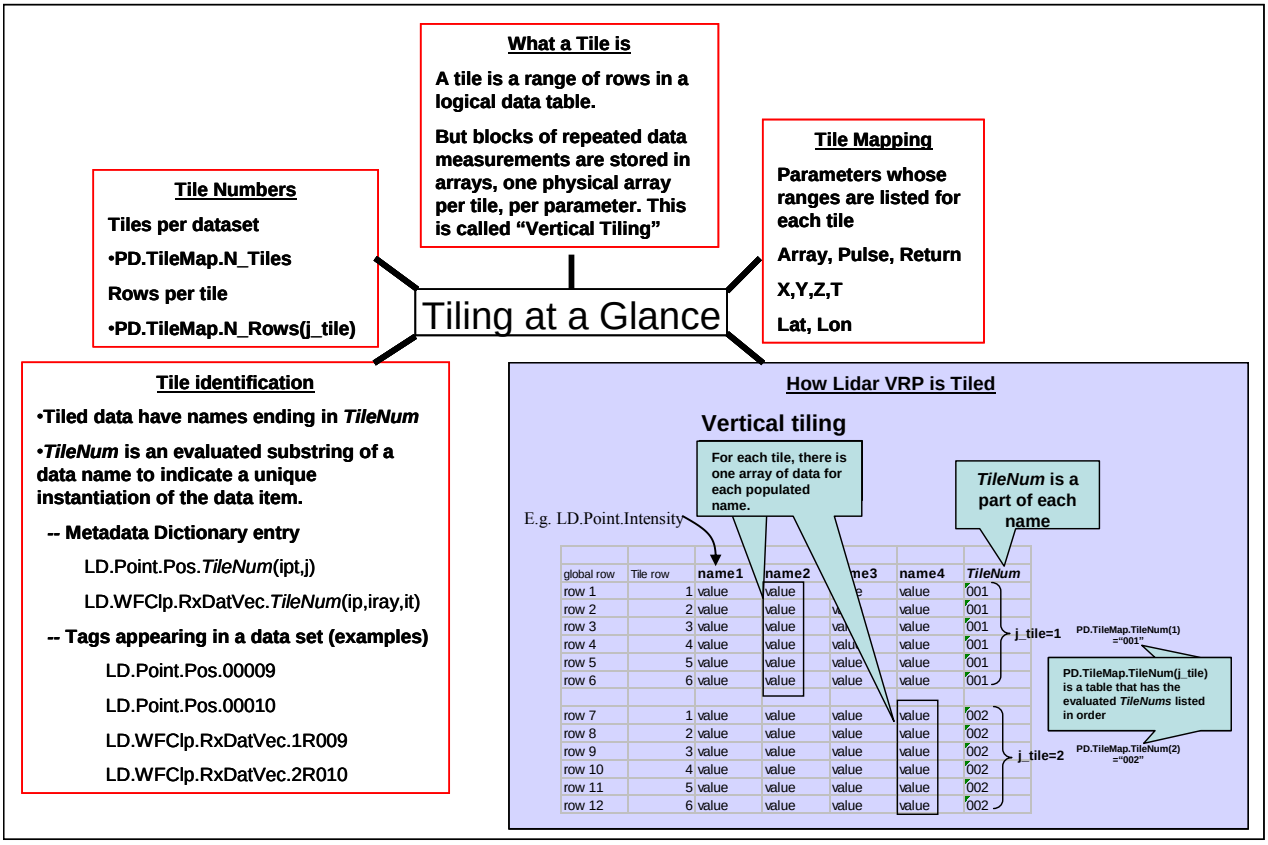


Figure 10.2.1- 1 Tiling's main features described at top level

10.2.1.2 Tiles and Name-Value Data set Population and Retrieval

To avoid the problems of brittle data architectures that flow from engineering via defining data in format documents and requiring data sets exchanged within an enterprise to comply with pre-conceived offset defined data, {"this is byte 245832, so according to the interface control document this must be the angle of the sensor handle in degrees} modern IT data structures build from self-documenting data files. The name-value construct is the most straight forward, transport-independent construct of self documenting files. The data sets have the names of their data entities stored internally and they can be accessed via software that extracts the value of a requested entity by name. This lidar conceptual data model requires the use of virtual[61] name-value transport files and interfacing with those files via name-value based operations[62].

So, if data is to be written to and extracted from lidar data sets on a name-value basis, tiles, which are the large data blocks involved in wideband data fetches need to have a unique name. The approach to this that permits a data definition document to anticipate varying file sizes and possible different data population plans is to have a computed part of the entity name for those items with multiple, logically equivalent, but value unique instances in a data set. For the clearest case in point, consider a point cloud. A lidar sensor could be generating data for a million or more points per second. The data for each point could be 32 or more bytes. A lidar data collection operation could go on for hundreds of seconds. So files are naturally multi-gigabyte scale. Observed in detail, the effectively 3-D images might span only 10s of meters on a display device and the processor might be registering adjacent scene components that represent only a percent or less of the scene at a time. But structurally, all the data chunks of the scene are equivalent. They are distinguished by the value of an embedded substring which we represent in CMMD documentation as "*TileNum*". Consider the position of a lidar data point. It is a 3 element tuple in a large array called LD.Point.Pos.*ChNum*.*TileNum*(ipt, j). The "LD" substring indicates that this is wide band **lidar D**ata. The "Point" substring indicates that this is associated with a point. The "Pos" substring indicates that the data gives an XYZ position. These are fixed fields in the entity data name. Every time a tag appears for the point position, it is literally spelled "LD.Point.Pos…". However, the italicized sub-strings, "*ChNum*"[63] and "*TileNum*" [64] are

[61] By virtual, we mean that via the accessing software, requests to post or extract data are on a name-value basis. The relative location of names and values in the actual transport file is not under control of the data definition document. Developers of the writing modules and matching reading modules may use existing formats and utilities such as HDF-5, or they may create internally indexed flat files along the lines of LAS2.0 or could actually create sequences of character names and binary or character values using reserved delimiters.

[62] The DAL is offered as a straight forward basis for implementation, but the DAL itself is not required for the CMMD data model to work. The name-value based data extraction is, however, required for the CMMD to be useable.

[63] Using evaluations of *ChNum* in LD.Point.Pos.* for a multichannel lidar is a unique strategy for level-2 data where the points are presented by channel. In level-3 and above, spectral and polarization channel points exist with single positions bearing channel specific intensity values.

[64] Rules for *TileNum* evaluated strings:

given computed values. The first instance (tile number) of a block of point data for channel 2 might be LD.Point.Pos.2.01 and the tenth instance of that point data would be LD.Point.Pos.2.10. Both are instances of an array indexed by (ipt, j). The point index, "ipt" distinguishes the multiple internal points, each of which have j=1,2,3 values associated with an X,Y and Z value. At level-2, different channels of point data [see LD.Point.Pos.*ChNum*.*TileNum*(ipt,j)], must have different values of *TileNum*.

Considerations for waveform clips

The convention of the lidar Conceptual model for enterprise level-2 data sets is that waveform clips will only be present in tiles that are dedicated to waveforms; as a rule, there will be no single tiles that carry both point vectors and waveform clip arrays. For data sets that have both waveform clips and lidar or synthetic lidar points, the evaluated *TileNum* strings that are for waveform tiles will be spelled with the initial two characters being "*N*R", where *N* is evaluated to a numeric character representing a return number from 1 to 7[65]. Tiles that begin with "*N*R" can cause sorting of waveform tiles into a separate subsequence from point data. That is consistent with the behavior when the data set is "points only". The "*N*R" prefix also results in sorting first returns first among the tiles. If point tiles are given purely numeric *TileNum* values, they will sort before waveform clip tiles. All these effects are consistent and complementary to a "points only" data set. This convention does not present a true "concavity" issue. The concavity issue is assuring that the design and practice of the tiling metadata definitions and usage avoid the problem that data that should be in a range is not, and is really in another tile. A situation could arise with waveforms that we have the same pulse-ray numbers in two tiles, one beginning "1R" and the other beginning "2R". How do we know that by having found a tile with the right range of pulse and ray indices, that we have all of the data? One answer that works is that a *fully qualified* query would specify that we seek waveform clips in tiles whose first 2 characters are "*N*R" under the rule just stated above. The full qualification of the query avoids the problem of failing to find the second return. Consulting PD.TileMap.WF_Returns(ia, j) prevents flailing by constraining the search to just the existing numbers of returns.

Also, we do not need to worry about having a data item that describes one returns sequence that has the points and clips combined in range sorted order. This is because the design of the returns code includes the information of how many lower returns are from points. Using the same code for clips as for points solves the problem because we can determine return number in

---

1) All values of *TileNum* in a single data set file must have the same number of characters. This number is documented in PD.Wordsize.TileNum.

2) The length of a *TileNum* substring may be chosen from 1 to 10 (TBR) characters.

3) Only alpha-numerics are allowed

4) All *TileNum* values must be assigned so that the tiles are consistently populated with the tile map metadata parameter values. If a dedicated tile strategy is in place (see 10.2.1.5), there may be restrictions on the names and ordering of tile numbers so that ranges of tile numbers correspond to the dedicated tile map.

[65] The maximum supported by the returns code scheme.

the presence of clips and points from a single PCD by point and clip separately. If the first return is a point and the second return is a clip, we can see that fact in ES.WFClp.RtnsCode.*TileNum*(iray,ip,j). For the converse case, we can determine that a point was preceded by some number of wave clips by looking at the code values in ES.Point.RtnsCode.TileNum(ipt). Note that a single return count can not be both a waveform clip and a point. This rule guarantees that the sum of wave clip returns and point returns is the total number of returns for a PCD. This rule is reasonable even when points are permitted to be defined as returns within the time range of an extended return that has a waveform clip provided. See figure 10.2.1-2

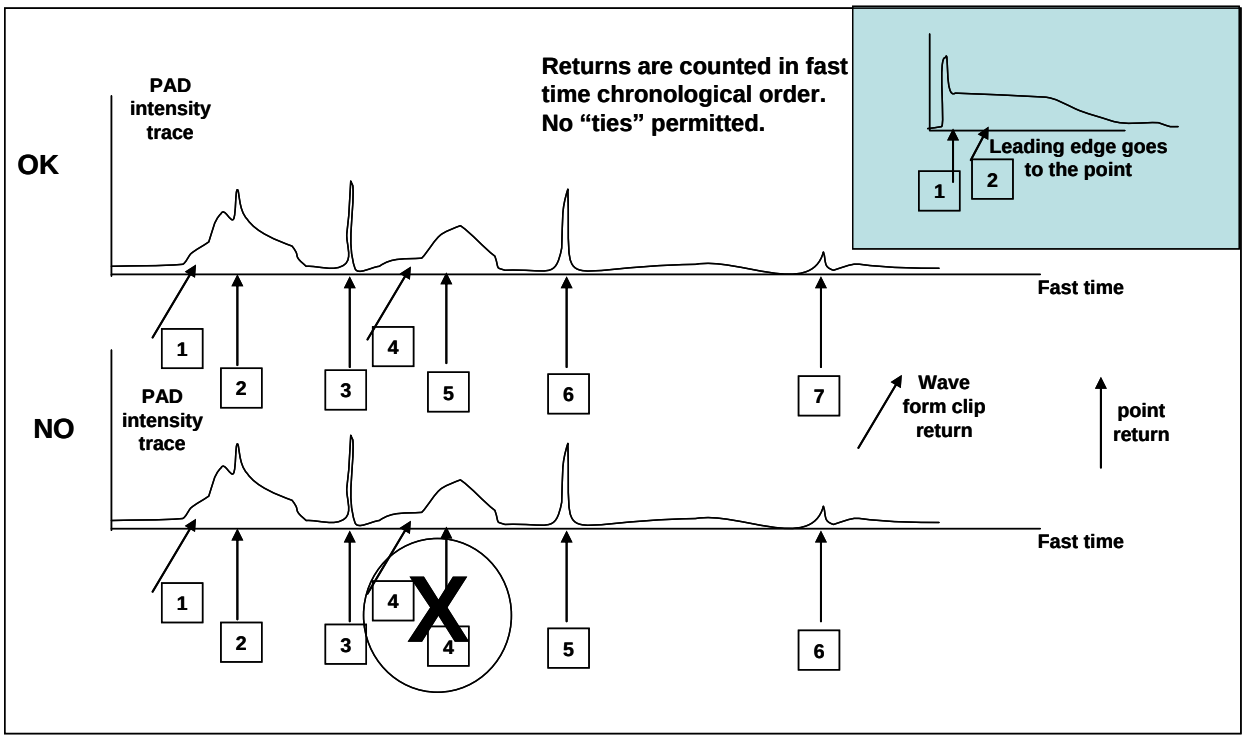


Figure 10.2.1- 2 Two return assignments for a single PCD trace.
"OK" upper diagram) Point return 2 seems to be a well defined time of a peak surrounded by an extended return. It clearly comes after the start of the extended return so numbering these 1 and 2 is sensible. Point return 5 is not associated with an obvious time to attach to the clip as corresponding to a particular point. But it comes after the beginning of the extended waveform so arbitrarily numbering these returns 4 and 5, while not justifiable, also produces no confusion in return counting. "NO" lower diagram) Point return 4 is given the same return number as the extended return in a waveform clip as it is embedded in. Thus return 4 is deemed both a point and a waveform. This assignment of return numbers is arbitrary, inconsistent and now the number of points+ number of clips> number of returns. Inset: While extended returns captured as wave form clips should be counted as starting at the beginning of their rise, a "tie" with a point embedded at the leading edge is broken in favor of the point.

Tile Structure Overview

An overview of tiling as applied to the lidar Conceptual model is given in Figure 10.2.1-3. Conceptually, the lidar wideband data is a logical table of many rows. Point associated rows (referred to in this document as "point rows") are each dedicated to a single "return" and consist of a position and a time (via pulse number) and optional applicable data unique to that 4-D measurement. Waveform clip data is also carried in rows, with the column headers defining the wideband data content associated with the fast time data records representing the recorded received signal for a pulse and a look direction (ray). Since a typical lidar collection consists of a tremendous number of pulses, even pulse associated data is managed more easily when it is stored by tile. A row of pulse data would consist of data associated with an individual and specific pulse. Column headers indicate the individual pulse specific items carried in the data.

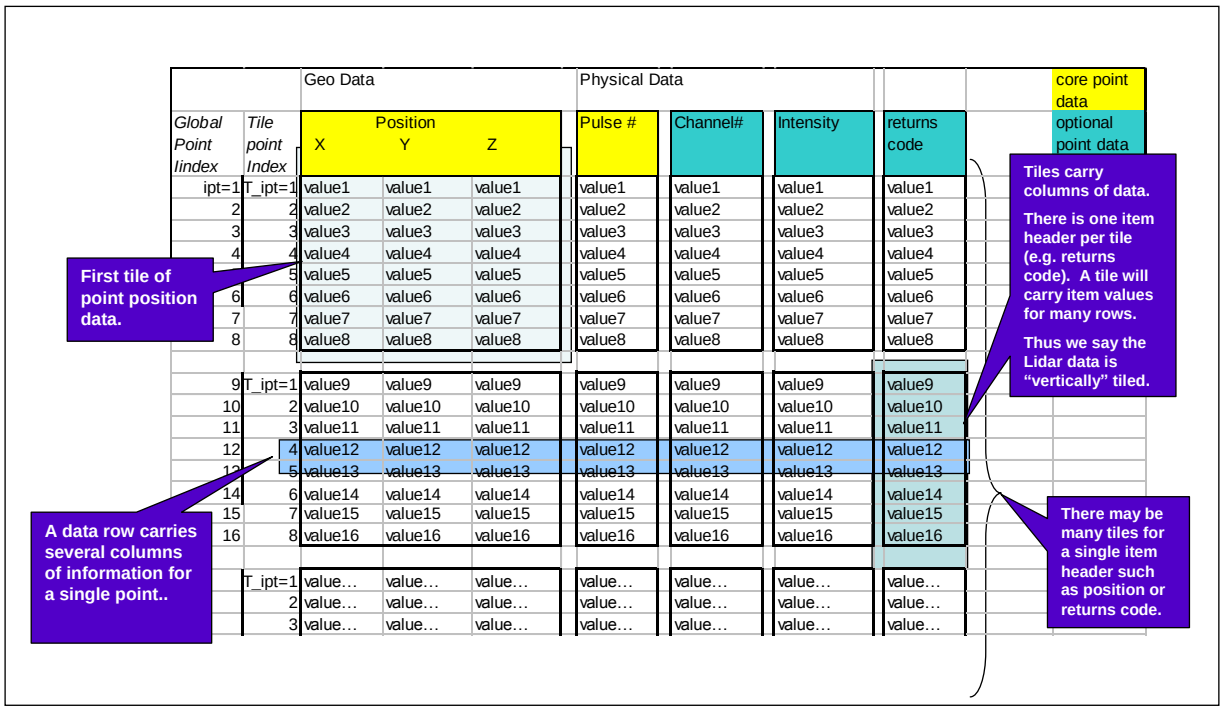


Figure 10.2.1- 3 Tiling Overview, Point Data Illustration

Vertical Tiling permits not only storing data in blocks that can be sized to usage needs for I/O management, but it supports retrieval of only data items of interest. The overall data set may be viewed as a collection of data rows (e.g. the "point rows" shown in this figure). The tiles store that data in separate columns. Wideband data is retrieved from a data set by tile. Tile mapping metadata support the capability to locate tiles containing data in a range of interest for selective transfer.

While the identity of lidar data in this conceptual model is along an association by row, we do not introduce a separate data item to be a row label or primary key. No meaning is associated

with the position of a row in the data set. In fact, it is expected that in use, data sets will be subject to operations that re-order rows and that they may be written to archival storage in these new sorting orders. A downstream processing or exploitation application may wish to create a temporary row oriented key which it keeps in memory, but there is no use for storing it with the data. [Imagine the column tile whose value in each row is the number of the row. That is a self redundant set of data.] Such an application can create this function without any additional support from the data model beyond the linking of the data within a row in the first place. Sorting of the rows of data by various column entries will preserve row integrity without appealing to an item in the data model that acts as a row identifier. The sorting software can create and delete such a data item without impacting the utility or meaning of the data.

That said, the global point index, "ipt", of Figure 10.2.1-3 just counts rows, continuing to increment from tile to tile. The tiles are stored by name and the data within them are sortable. Thus there is an implicit, but largely irrelevant row ordering in the global data set. The tiles appear as arrays, and within each tile the row index, "T_ipt" for this discussion, can be used to identify a row locally. Since reconstruction of a complete row requires the retrieval of data from multiple vertical tiles that have the same globally referenced rows, one tile for each "column heading", it may appear at first glance that without the "ipt", we can not be assured of combining the same global row index data from all of the vertical tiles to reconstruct a full row. This is not true, however. The method used in this lidar conceptual model is to include a common tile identifying string in the <u>names</u> for the various column tiles. By having the same data rows (taking the *kinds* of tiles, i.e. points, pulses and waveform clips separately) in all tiles containing the same value of *TileNum*, proper row correspondence between column contributions from the tiles can be assured. Working from only point tiles of the same value of *TileNum*, data from the same T_ipt[66] is assured to be for the same row. More detail on this is addressed in 10.2.1.5 under the <u>internal tile mapping</u> heading.

Let us consider a tabular representation of key lidar data content presented in Figure 10.2.1-4. The upper box portrays several rows of data associated with a single lidar point each. Every row has a number of columns that represent data items that can be populated. The core point data items that will always be populated are the XYZ position components and, frequently, the pulse number. In addition, possible additional items, depending on the mode and sensor design, include the light intensity and a returns code. Let us imagine that a single data set has 10,000 such point rows for each pulse. Also let us imagine that each scan involves 100 pulses, and that the sensor scans at a rate of 20 Hz. Thus the PRF is 2kHz. For a data set covering 600 seconds, that corresponds to 120 billion rows. Processing might want 200 pulses at a time. So, 2

[66] Since the row index is effectively a dummy index, the detailed paragraph sections will use the symbol, "ipt" for point row index. This is a shorter symbol than T_ipt, and although in those paragraphs, the usage for an array from a specific tile is consistent with the term, "T_ipt", used in the present discussion, "ipt" will be used and interpreted as a local tile based value that can be related to the true, global, ipt index value via the Tile mapping parameters. There is such a rare need to do that, however, that no shortcuts are provided.

million rows at a time might be a natural I/O group request. Rather than ask for 2 million reads to ingest that amount of data, or getting all 120 billion rows in a single massive, system hanging, I/O operation, we can get 2 million rows of the point data in 4 reads of 1 tile (number) each for the position, pulse number, intensity and returns code. The request for the first such data ingest would use LD.Point.Pos.*ChNum*.00001, ES.Point.PulseNum.00001, PF.Point.IrayNum.00001, LD.Point.Intensity.*ChNum*.00001, and ES.Point.RtnsCode.00001 in calls to the reading module of the DAL. To access data from near the middle of the file from the full collection operation, the requested tiles might be LD.Point.Pos.*ChNum*.29901, ES.Point.PulseNum.29901, PF.Point.IrayNum.29901, LD.Point.Intensity.*ChNum*.29901, and ES.Point.RtnsCode.29901.

Looking at the middle box of Figure 10.2.1-4, we see that there is a lot of data associated with individual pulses as well. The names of tiles containing the pulse linked data include SC.Pulse.TxCount.*Il.TileNum*(ip,j), SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip, j), SC.Pulse.SensorAxisX.*TileNum*(ip,j), SC.Pulse.SensorAxisY.*TileNum*(ip,j), SC.Pulse.RecordStrt.*TileNum*(ip,j), SC.Pulse.Scan.Num.*TileNum*(ip,j), SC.Pulse.Scan.Phase.*TileNum*(ip,j), SC.Pulse.ArrayMajorAxis.*TileNum*(ip,j), SC.Pulse.BmMajorAxis.*TileNum*(il,ip, j), SC.Pulse.PolCode.*TileNum*(ip,j), and SC.Pulse.FilterCode.*TileNum*(ip,j). The sub-field, "*TileNum*" in each of those data tags is to be evaluated and the full data tag used in the read module calls.
The volume of data at the "array" level, shown in the bottom box is minimal, so the advantage of blocking the data for I/O is not so great. We use ordinary arrays for those items. On the other hand, waveform clips may involve many thousands of samples per clip, creating a data volume problem that can be generally managed with the tile concept.

Waveform clips present a slightly different problem than point and pulse data. The 'width' of the columns is disparate, with the intensity time history being thousands of times the data of the other columns in a waveform row. If we were certain that we would only have a few waveform clips in a data file, it could make sense to only tile the intensity. But, if we wish to let the clip have a possible size of the full receive window of data for a pulse, and we wish the same scheme to work at level-1 as we might use at level-2, then a full tiling is necessary. There could be as many waveform clips as the number of pulses in the collection times the number of elements in the arrays, that is, many millions. So, even if the clips were very short, the data must be defined within tiles. What happens is that we size the number of rows in a tile to the intensity data load and the matching tiles for other parameters like the number of samples, return code, pulse number, and array element are just lower in byte count. The tiles for carrying waveform clips have the names, LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), ES.WFClp.RtnsCode.*TileNum*(ip, iray), and LD.WFClp.Tstart.*TileNum*(ip, iray).

Organizational data		Geo Data			Physical Data						core point data
Point Identifier	*tile*	X	Y	Z	Pulse #	Intensity	returns code				optional point data
ipt=1		value	value	value	value	value	value				
2											single measurement information group
3											
…											
Pulse Identifier	time	sensor pos	sensor orientation	RecordStrt	Scan#	scan phase	scan major axis angle	pol code	band code		core pulse data
ip=1											optional pulse data
2											
3											
…											
Array Identifier	pol code	band code	RayDirs.*Ia* (ie ,,j)								core array data
ia=1											optional array data
2											
3											
…											

Figure 10.2.1- 4 "Row" model of point, pulse and array [determined by the channel description] associated data groups

Figure 10.2.1-5 indicates how rows in pulse oriented tiles are indexed to correspond to unique values of the channel (via the combination of specifying a pulse and a ray).

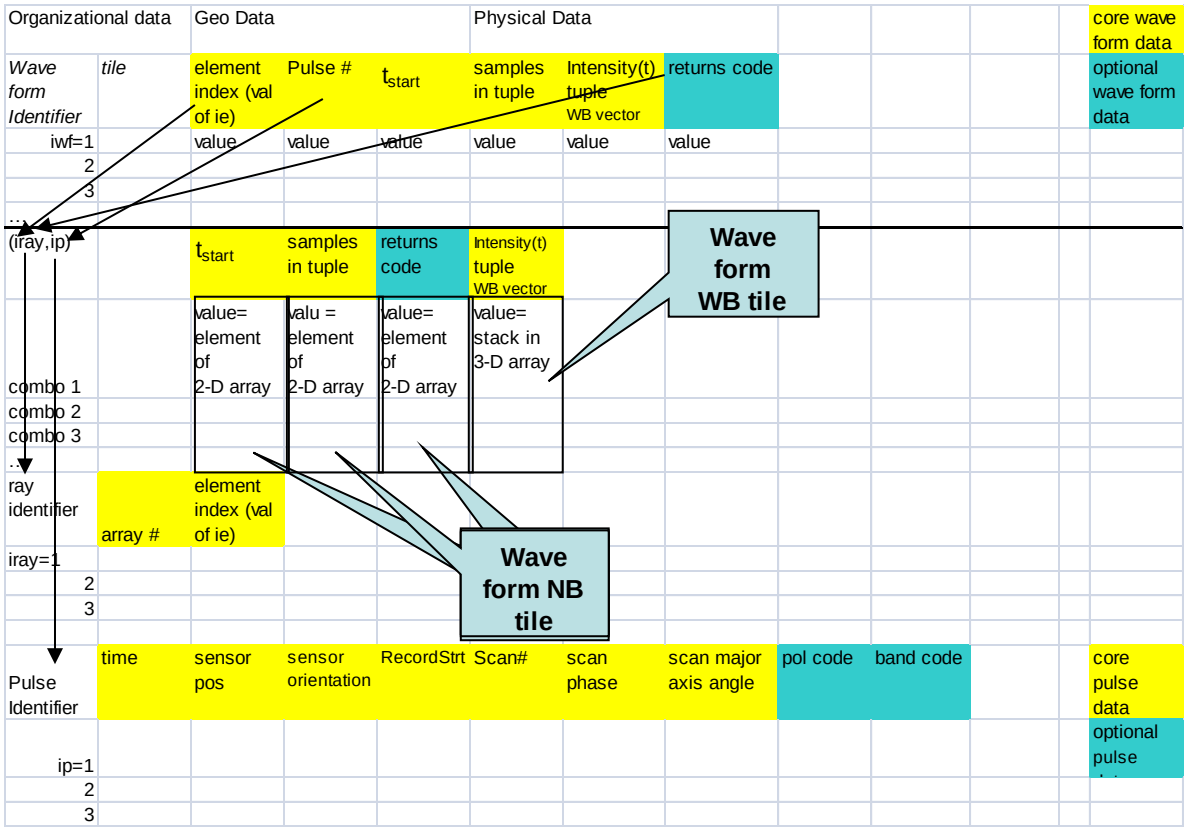


Figure 10.2.1- 5 "Row" model of waveform, pulse and array associated data groups
Logical data row can be expressed as a data array by replacing iwf, the notional waveform row index illustrated above the bold line in the diagram, by the pair, (ip, iray) for pulse and ray indices. The ray index, in turn, is the pair (array, element id). This data definition document does not employ a waveform index, "iwf"; rather it uses "ip" and "iray" to support consistency between enterprise level-1 and level-2 waveform treatment. Breaking the pulse out enables data sets with enormous numbers of waveforms to more readily address the data within tiles.

10.2.1.3 Tile Sizing

We note first that tile sizing is chosen in the data set writing stage. This may be in response to direct input requests to the lidar data set writer at the particular product level being generated, or the writing application may be endowed with models of the target processing hardware and able to compute good tile sizes from some basic parameters. This data definition document does not need to prescribe how the sizes are arrived at, only how they impact the population of the data in the products.

That said, there are some considerations that affect tile sizing decisions that help understanding of the lidar data conceptual model, particularly for enterprise level-2. Level-2 includes stage-1 (pulse level transformation to XYZ, either in sensor reference or earth reference.) as well as stage -2 where data has been registered into scans and may include both waveform clips and points. A consideration is the total size of the tile as well as the spatio-temporal coverage of the tile. The amount of waveform data in a region can make it so that byte volume considerations outweigh 4-D extent preferences. The choice of tile bounds is made in writing the data sets. It can always use the more limiting of byte volume or coverage[67]. **It is advised that tiles be sized to composite to scan boundaries whenever practical**. Note that subswaths may have overlaps that are artificial due to the processing algorithm creating data redundancy to support registrations[68]. Thus, the highest logical data division for assured non-redundant tile coverage is scans. Those repeated points resulting from processing subswath assignments should not be replicated in multiple tiles. Collection subswaths can contain complete, non-redundant scans, and following the scan rule could result in collection subswaths compositing to tiles. However, other considerations may outweigh the collection subswath composition to tiles, such as wanting a simple geographic boundary to the tile that is well represented by the envelope data in PD.TileMap.Lat/Lon.

10.2.1.4 Tile Alignment

Note that we have effectively introduced tiles to block 3 types of data:

- Data associated with points
- Data associated with pulses
- Data associated with waveform clips.

While it is not difficult to support reuse of *TileNum* string values between point and pulse oriented tiles, allowing this as an option would require another metadata item or added CONOPS complexity [such as DAL specific linkage of the capability], force all implementations to have the alternatives coded and have no reasonable benefits. So, we make the rule for *TileNum* strings be that no two j_tile values can have the same value of *TileNum*. Further, within a single type of data to be tiled, all of the tiles need to have the same mapping into rows of the data diagram as represented in Figure 10.2.1-1. So, if LD.Point.*ChNum*.Pos.00571 covers what might be rows 1,142,000,000 to 1,144,000,000 in Figure 10.2.1-1, ES.Point.PulseNum.00571 must also cover rows 1,142,000,000 to 1,144,000,000.

Another impact of the convention of the preceding paragraph is that at level-2, different channels of point data, must have different values of *TileNum*. A multichannel lidar at L3 will have points, which may or may not have been created from data derived from different pulses.

---

[67] When the waveform clip volume permits, some correspondence of *TileNum* values for points and waveforms for the same general scene volume can be built in. E.g., 1N001 and 00001 could be *TileNum* values for waveform and point tiles with returns in the same vicinity.

[68] This refers to using the single "row " of the point data as a member of more than one grouping of points to facilitate registration.

Nonetheless, we can assign a pulse number to such a synthetic point. When all the data contributing to a multichannel point is from the same pulse, the pulse number reporting is clear. When multiple pulses are involved, the pulse number should report the middle pulse or the earlier (arbitrarily chosen over later) of the two middle pulses in the point synthesis input point set. There will be no *ChNum* substring for L3, L4,L5 multichannel points and the issue of tile number and channel goes away.

Another rule must be followed to assure that the tiles can be efficiently mapped. There is an assignment of the values of *TileNum* to an ordinal count of tiles that is supplied in PD.TileMap.TileNum(j_tile) where j_tile ranges over all of the tiles in a data set. All point related tiles must be in a contiguous sequence of j_tile. All waveform bearing tiles must be in a separate contiguous sequence of j_tile and all pulse oriented tiles must have their own contiguous sequence of j_tile values. When a data set has no points or no waveforms, there would be no subsequence of j_tile for the non-existent data bearing tiles. Substantial capability accrues to populating the pulse tiles, so those should generally be present in most lidar data data sets.

Overlaps for Tiles

When local Cartesian frames are aligned with tangency to the ellipsoid, adjacent tiles will have data that is presented in adjacent rectangular parallelpiped 3-D volumes that are tilted relative to each other. If they butt at the surface of the ellipsoid, there will be gaps at elevations above the ellipsoid. If they butt at the top of the 3-Volumes, there will be overlap at lower elevations (figure below).

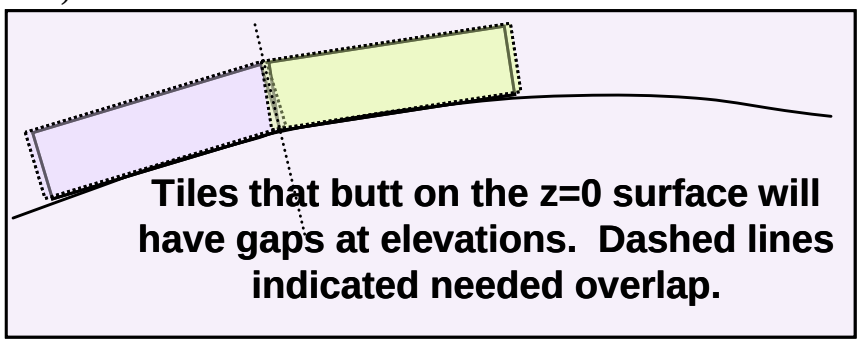


Figure 10.2.1- 6 Overlap is needed to avoid introducing voids when tiling to local geodetic geometry.

To avoid having any regions of tile overlap, one would avoid ES.Align.XYZ_Flag=2 or 3. One should favor a value for this flag of 4, or possibly,1 in the case that the data set is restricted to a single UTM zone. Since the CMMD could not be permitted to prohibit collection across UTM zone boundaries, the options for Cartesian local geodetically aligned and global frames were included. The local geodetic alignment introduces the need for overlap and need not be used if the disadvantages of overlap in the use case outweigh the benefits of local geodetic alignment.

The data definition permits the use of XYZ MGRS referenced coordinates[69] or arbitrary XYZ based on ECEF orientations of the basis vectors. The MGRS zone is assigned one to a tile, although it is permitted that multiple tiles have the same MGRS zone assignment. When MGRS is used for scan composited tiles, any scans that cross a UTM boundary present a tiling issue because some data will wind up having invalid[70] UTM XY values. The following objectives drive the adoption of a few simple rules to govern this situation.

- Objective 1: The data definition must allow that processing not be forced to create voids where there are no collection voids.
- Objective 2: The data definition must not force duplicate collection near MGRS zone boundaries to avoid producing voids.
- Objective 3: Complete scan bounded tiles must always be a permitted tiling choice.

To assure that these objectives can be met while following the tiling and metadata rules of the data definition the following rules must be adopted:

20.2.1 Tiling options are available that do not use both duplicate point population and overlap to avoid tiling void artifacts. However, overlapping tiles can have duplicated points with no negative impact to tile searching and point display. Duplicate population should be used to prevent artificially creating voids or creating a need for complex additional metadata. Duplication of invalid points as valid points in an adjacent tile is likewise benign. The volume of data in overlapping regions will be minimal if the minimum overlap required to assure artificial void avoidance is employed.

20.2.2 To avoid the creation of voids by processing, any scan nominally[71] covering more than a single UTM zone must be processed in each tile and MGRS zone for which it is to be output. This means that scrap ends of scans protruding into adjacent MGRS zones with no tile coverage do not need to be processed in the valid zone, but that duplications are needed if there is tile coverage in the valid zone.

20.2.3 All valid points should be retained in their respective tiles.

To be clear, the points in the Cartesian frame point cloud are always valid with respect to geolocation in ECEF, and consequently geodetic coordinates. The term “invalid” signifies only that the horizontal UTM coordinate for the point is out of range for the MGRS zone indicated for

[69] Note that MGRS zones are not coordinates themselves. But we could make the southwest corner of a zone the origin of a tile Cartesian frame and orient the axes along zone edges and then we would say the coordinates are MGRS referenced.

[70] In our discussion of UTM related issues, the term “valid” means that a computed position within a UTM zone has as coordinate value that remains within the UTM zone. “Invalid” means that the position is within the range of another, usually adjacent, UTM zone. UTM zones are Cartesian grids that piecewise fit local facets of the WGS-84 ellipsoid. They are sized to permit the local Cartesian assumption to be sufficiently accurate at the meter level over a zone.

[71] The term “nominally” is used to indicate that the determination of the zone that a point falls in for UTM frame XYZ computation does not assume any geolocation error contribution, but rather is computed as if the point computation is error free.

the relevant tile. This is an informational point only and in no way represents a limitation on the utility of the points in the data set.

Note that tiles have both temporal and spatial bounding data defined. This permits dwelling modes to tile by pulse number, but spatial tiling is permitted for such modes as well.

10.2.1.5 Tile Mapping

External Tile Mapping

"External" tile mapping refers to describing the relationship between tiles and how the data in the various tiles complement each other to compose the over all data set. Topics in external tile mapping include tile naming, identifying tile sizes and the range of content in the individual tiles. External tile mapping supports efficiently finding particular data or data ranges within a large data set.

The problem of tile mapping is one of supplying the metadata table structure that permits one to determine which tile(s) has the data that is sought. The data sought will lie in some range of 4-D spatial-(slow)temporal parameters as well as in a range of values for any other point, pulse, or waveform clip row parameter such as return, intensity, channel (spectral band or polarization), sensor position, scan phase or look direction. The PD.TileMap.Lat/Lon data items provide the latitude and longitude horizontal extents for each tile. While this may appear to provide closure of the tile mapping problem, there are some remaining issues.

It is expected that if a lidar has more than a single array so that SC.Array.* is populated for more than one value of "ia", users may wish to retrieve points on a channel basis. This is readily supported by making single channel tiles [perhaps by sorting first on array number and then on spatial or intensity parameters, followed by tile allocation by array number]. Proper search qualification for point data would then include a specification of the channel of interest for the specific data retrieval request. At level-2, tiles are channel specific, so even the case of spatially diverse channels does not pose a tile mapping problem . However, a similar situation would make use of the tile maps very difficult. We discuss that next.

We introduce the term, "fully qualified query" to include all constraints, particularly specification of an array via a spectral band or polarization constraint. This is especially important when tiles are dedicated to single channels. We intend that when a fully qualified query of the tile map indicates a particular tile or set of tiles has the desired data, no other tiles should need to be searched for the required data. This means that no data within the outer bounds of a tile's range should be in another tile. Fortunately, we can assure this by adopting a rule set regarding sorting and sub- and up-tiling. The rules of 10.2.1.1 prevent "concavity" of data in tiles. Concavity would lead to indication that data is in a tile when it is actually in a different tile. To illustrate, suppose that we permitted sub-tiling of a tile that is sorted first by Z. We may want the data corresponding to a range of 20,000 meters X and 10,000 meters Y to 30,000 meters X and 30,000 meters Y in a particular MGRS zone. But the data may have been sorted with Z= -10 meters to 50 meters in one tile and Z=51 meters to 200 meters in a separate tile. The ground may be sloped, as in the case of a hill so that the high ground is a central region in the data set when shorted by horizontal position. But splitting the tile after ordering it by Z

creates a low Z tile and a High Z tile. The low Z tile may have bounds for XY and Z as requested, but the middle of that region is in the high Z tile. One could not correctly conclude the sought horizontal range was in the first tile found with the XY bounds of interest. Thus, for the tile map to support efficient tile identification, the bounding data is supplemented by enforcing rules in the data definition. It may be very advantageous to make Z the primary sorting parameter for some applications. This is permitted. But the user doing the Z sort must properly update the sorting priorities in the metadata so that a subsequent user, who must consult the sorting priority metadata as part of his process to create sub-tiles, does not unwittingly create a concave tiling.

A fully qualified geographical request should not include a Lat-Lon range, as that could lead to crossing tile boundaries. Rather, a request specified by a (presumably central) point will result in a finite list of candidate tiles. {Overlapping tiles might result in a point being within two or four tiles}. A large search region could result in many more than four tiles having portions of the requested region. Given an otherwise fully qualified request, any of the tiles identified by a single-point geographical request satisfies the request. Tools can be devised that support query by tile directly and which support returning a list of tiles with the requested point within their bounds.

This data definition document for lidar provides documentation of the original global sort that existed in the data previous to its original tiling (see PD.Sort.Native*). This remains available[72] after any number of sub-tiling, sort and up-tiling operations have been applied. The consequences of any sorting done to point data tiles after their original creation are documented in PF.Sort.PointPriority.*TileNum*(j). For the pulse oriented tiled data, the corresponding sort information is in PF.Sort.PulsePriority.*Tilenum*(j). *The point is that with a tile map, if we want some range of data, we know where to find it and don't have to ingest more than the right tile(s).* Also note, that if PD.Point.RangeListC.*OrdIndx*(j,k) and PD.Point.RangeListX.*OrdIndx*(j,k) were populated previous to sorting any data, these tables need to be recalculated to reflect any changes in the mapping of pre-computed error ellipsoids to the points in the point cloud.

Dedicated Tiles

Some parameters, particularly the channel number, the return number (this CMMD uses 100 to designate "last" regardless of actual return count), and in the case of level-4 data, the sensor number, are naturally suited to having dedicated tiles. By this we mean that we would have cause to populate a tile with points that only come from a single channel (a level-2 unique option) or from a single channel and a single return (most of interest would be first and last). For a single band/polarization-single quench Geiger mode lidar, all tiles are dedicated by default and need no additional documentation to determine which band or which returns are in which tiles. But for more diverse data collecting lidars, we need a method of saving software from having to

[72] Within the rules of the CMMD. -- To have the data present, the software that writes files will need to populate the fields.

examine the data for every tile in order to determine which tiles are for a given channel or return when the tiles have been purposely populated in a dedicated fashion. The application that did the sorting of data into the tiles can document the results of that process with a small number of parameters. We generously limit the number of dedicated parameters in a single data set to 3 and supply several parameters that could be interpreted as populating a table such as below.

1st Constraint parameter e.g. channel#	2nd Constraint parameter e.g. return#	3rd Constraint parameter e.g. sensor or coll'n#	TileNum Low	TileNum Hi
1	1		0001	0500
1	100		0501	1000
2	1		1001	1500
2	100		1501	2000

Figure 10.2.1- 7 Illustration of documentation of dedicated tile groups. Each row of the table describes a tile group. Each column is provided by a PD.TileMap item.

This table represents the case where two parameters are used as dedicated tile parameters: the channel number for two channels and the return number (where we sort the data into first and last and discard all other data for this data set, if it existed). There are 2000 tiles and each tile category/group (channel 1, first return), (channel 1 last return), (channel 2, first return), (channel 2, last return) each have 500 tiles. We use a 4 character code for *TileNum* and encode the tile number substring with the sequential number of the tile. To represent this table we populate the following:

PD.TileMap.DedictdTile.Cnstrnt1.Name
PD.TileMap.DedictdTile.Cnstrnt2.Name
PD.TileMap.DedictdTile.Cnstrnt3.Name
PD.TileMap.DedictdTile.Cnstrnt1.Val(j) j=LoIndx to HiIndx [1 to number of dedicated tile groups]
PD.TileMap.DedictdTile.Cnstrnt2.Val(j) j=LoIndx to HiIndx [1 to number of dedicated tile groups]
PD.TileMap.DedictdTile.Cnstrnt3.Val(j) j=loIndx to HiIndx [1 to number of dedicated tile categories]
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j)

Note that we have grayed out the third constraint parameters since we do not need to populate them in this example.

Note that tile groups must be consecutively numbered so that they can be defined by first and last tile numbers only. The use of dedicated parameter tiles does not cause "concavity", but it does mean that ranges of spatial and temporal coverage are repeated in each of the dedicated tile

sets. This should be considered normal and since it is always documented by metadata, software does not have a dilemma like it would face in the "concave" data distribution across the tiles. Data sets that have this kind of sorting across tiles are tailored to uses where the data in tiles would be used independently or superposed in a viewer. If that arrangement is not suited for an application, a more natural tiling should be generated in a separate task. The dedicated tile parameters might be queried as part of the data set selection process to assure proper processing tasking and data set selection.

Note also that when an explicit dedicated tile strategy is used, the point row data for the dedicated item does not need to be populated since it can be derived[73] from the dedicated tile data PD.TileMap.DedictdTile.*.

Further note that population of PD.TileMap.PtType(j_tile) makes a tile explicitly dedicated without recourse[74] to **PD.TileMap.DedictdTile.Cnstrnt*n*.*.**

The above discussion focused on point oriented tile issues. There are pulse oriented tile issues as well. As indicated in the channelization discussion of 10.1.3.4, a subtle interplay between the channelization description scheme and the tiling scheme used in this CMMD exists so that the number of metadata items and dimensions of data arrays can be kept to a minimum. Pulse oriented tiles need to be constrained to a single channel when used to support documentation of IPP sequences that are distinct for different channels. This is done via the dedicated constraint construct for tiles and a single value of *TileNum* for pulse oriented tiles can only apply to a single channel. Channels due to 2 lasers and one array will need separate IPP sequences. Channels using independent laser and array pairs might need separate IPP sequence data for SC.Pulse.TxCount, but for best accuracy, should supply at least the SC.Pulse.TxDelay data. Channels that use separate arrays or filter wheels may have a common IPP sequence. So we won't populate pulse tiles redundantly when the same IPP sequence is used for multiple channels. When there is only one pulse sequence in a data set, but there are multiple channels, the single IPP sequence provided should be used for all channels. An error condition could be detected by determining whether there are more lasers in the channels than arrays when there is only one IPP sequence. Such a case would mean that the data set should fail a validity test.[75]

Geographical Tile Layout on a grid

A typical need for an exploitation application is to identify the identity of tiles adjacent to a given tile and in particular directions. For example, a display or processing tool is displaying or working on the point cloud over particular tile's worth of horizontal extent. The user wants to scroll up and left on the screen from this tile. How does the software know which tile to fetch

[73] So, dedicated tiling can be used as a data volume reduction strategy.

[74] Because we are not permitted to mix real and synthetic lidar points in a single tile.

[75] Because two lasers cannot fire at the same time and illuminate a single array while maintaining channel independence for the recorded data.

next? This task is supported simply by three data items: PD.TileMap.AdjacencyPts(n_tile), PD.LoIndx.AdjacentP(j), and PD.HiIndx.AdjacentP(j). The Hi/Lo Index data indicate the number of tile rows and columns in the geographical tile grid. The AdjacencyPts data indicate which row and column positions have been populated with actual tiles of data. This combination supports products where the actual data tiles do not composite to a filled rectangle of point cloud data. The detailed discussion paragraph for PD.TileMap.AdjacencyPts(n_tile) gives the expressions for determining the names of target tiles on the tile grid in terms of a current tile and and the distance in rows and columns to the next tile of interest. The same scheme with slightly different metadata names ("W"s in the metadata tag) supports selecting wave form bearing tiles.

Internal Tile Mapping

"Internal" tile mapping refers to describing the representation of data rows within the individual tiles. Principally, the internal mapping metadata have been defined to facilitate data volume reduction and the association of tile rows with data rows for point and pulse oriented data. Topics in internal tile mapping include associating position within a tile with position in the greater data set and handling redundancy.

Since we have adopted the "vertical" tiling data design, all elements of a tile are the same parameter. We need to provide a value for each "row" of data in the conceptual model. Let us define a "logical" row to correspond to a unique row in the database view of, for example, figure 10.2.1-4. However, some tiles have some values that are the same for each logical row, indexed by "T_ipt" in the tile overview discussion of section 10.2.1.2. This is certainly true for dedicated parameter tiles. For some non-dedicated parameters, the internal sorting may result in lengthy stretches of rows that have the same value of a given parameter. We avoid wasteful redundant data population via a flag and a number of repeats indicator. But that means that we need to be able to compute the logical row index, T_ipt range, from the actual row index in the tile and the actual row index from the logical row index. The internal tile map data enable this. Basically there are two cases. Either the data has enough consecutive repetitions that indicating what the repeated value is and how often it occurs is worthwhile or it is fine to just write the data (column vector) out element by element so that the actual row index is also the logical row index. The application that writes the tiles will know which case applies. To communicate that to a reading application we use a flag associated with each named tile. The names of the internal mapping data flags are:

PD.IntTileMap.RtnsCode.*TileNum*.Flg
PD.IntTileMap.PulseNum.*TileNum*.Flg
PD.IntTileMap.ArrayNum.*TileNum*.Flg
PD.IntTileMap.ChNum.*TileNum*.Flg
PD.IntTileMap.PolCode.*TileNum*.Flg
PD.IntTileMap.FilterCode.*TileNum*.Flg
PD.IntTileMap.RecordStrt.*TileNum*.Flg
PD.IntTileMap.Scan.Num.*TileNum*.Flg

PD.IntTileMap.Scan.Phase.*TileNum*.Flg
PD.IntTileMap.Pos.*TileNum*.Flg
PD.IntTileMap.Intensity.*TileNum*.Flg
PD.IntTileMap.Nsamps.*TileNum*.Flg
PD.IntTileMap.TruthTags*TileNum*.Flg
PD.IntTileMap.Tstart.*TileNum*.Flg
PD.IntTileMap.OriginX.Flg
PD.IntTileMap.OriginY.Flg
PD.IntTileMap.OriginZ.Flg

When a PD.IntTileMap.*.Flg is set to the value, 0 (which is also the default value[76] when there is no population of the flag), the corresponding tile is an ordinary array and the actual row index equals the logical row index. When a PD.IntTileMap.*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile and the true number of rows in the array (as opposed to the number of logical rows) is provided by PD.IntTileMap.*.Flg. To support this methodology for native 1-D data, we have to add an index, j, to the tiled data array. In the default case the range of j is 1 value, making the array degenerate to 1-D. When we need to use the repeat factor trick, the range of j is 2 values so that we can carry a repeat factor and a data value in a column each. In the native 2-D data case, we re-index in 2 dimensions so that there is effectively a row for each value and a repeat factor for that value.

The following flags, although conceivable, are not defined due to the fact that with reasonable sorting priorities, this data by its nature does not repeat:

PD.IntTileMap.TxCount.*TileNum*.Flg
PD.IntTileMap.SensorPos.*TileNum*.Flg
PD.IntTileMap.SensorOrntn.*TileNum*.Flg
PD.IntTileMap.Scan.MajAxisAngle.*TileNum*.Flg
PD.IntTileMap.Intensity.*TileNum*.Flg

The tile arrays affected by the internal mapping flags are:

ES.Point.RtnsCode.*TileNum*(ipt,j)
ES.Point.ChNum.*TileNum*(ipt, j)
ES.Truth.Tags.*TileNum*(ipt,j)
ES.Origin.X*
ES.Origin.Y*
ES.Origin.Z*
ES.Point.PulseNum.*TileNum*(ipt,j)

[76] This statement is to be interpreted as direction to the writers of CMMD compliant software to initialize these parameters to zero.

PF.Point.IrayNum.*TileNum*(ipt,j)
SC.Pulse.PolCode.*TileNum*(ip,j)
SC.Pulse.FilterCode.*TileNum*(ip,j)
SC.Pulse.RecordStrt.*TileNum*(ip,j)
SC.Pulse.Scan.Num.*TileNum*(ip,j)
SC.Pulse.Scan.Phase.*TileNum*(ip,j)
LD.Point.Pos.*TileNum*(ipt,j)
LD.Point.Intensity.*TileNum*(ipt,j)
SC.WF.RayDescElmt.*TileNum*(iray)
ES.WFClp.RtnsCode.*TileNum*(iray, ip)
PD.WFClp.Nsamps.*TileNum*(iray, ip)
LD.WFClp.Tstart.*TileNum*(iray, ip)

Default case: 1-D data vector

This section describes the use of repetition factors for those tiles whose default data is a 1-D vector (think of a column of data values). For each "non- WFClp" data item above the default data is effectively a column vector, and (other than LD.Point.Pos.*ChNum*.*TileNum*(ipt,j), the range of j is from 1 to 1 when the flag is 0 and from 1 to 2 when the flag is greater than 0. In those cases, the j=1 entry is the number of times the value on that row (given value of the first index) occurs consecutively. If the value appears again after a different intervening value appears in a row, a new real row in the tile has the repetition value for that later logical row sequence as a later separate entry in the actual tile array.

For LD.Point.Pos.*ChNum*.*TileNum*(ipt,j), the range of j is from 1 to 3 when the flag is 0 and from 1 to 4 when the flag is greater than 0. For the flag =0 case, j=1,2,3 corresponds to X,Y,Z. For the flag equal to 1 case, j=1 slots carry the number of logical row repetitions and j=2,3,4 correspond to, e.g., X,Y,Z data.

To illustrate, consider a two channel lidar with data sorting so that a tile has 10000 rows with alternating values of 1 and 2. The tile and flag would appear as

flag =0 . But if the data were sorted on channel# we could have **flag=2.**

value
1
2
1
2
1
…
…

n	**value**
5000	1
5000	2

A 10000 byte tile with flag=0 could be represented in 8 bytes in the illustration above. When there may be thousands of such tiles in a data set, this approach could be valuable. The position tile, LD.Point.Pos.*ChNum*.*TileNum*(ipt,j), for the same *TileNum* could well have 10000 real rows in either case. There is no problem with there being different actual tile row counts for

corresponding tiles for columns in a set of data rows as long as the logical row counts are consistent.

Actual row value (row) from logical row (KL) :

```
sum=0, row=0
Do While (KL>sum)
   row=row+1
   sum= sum+ PD.*.TileNum(row,1)
End do
Value[77]= PD.*.TileNum(row,2)
```

Logical row range(KL1,KL2) from Actual row (row) :

R_0=0, R_k=PD.*.*TileNum*(k,1) {k ranges over first to desired actual row}

$$KL1=1+\sum_{k=0}^{k=row-1} R_k$$

$KL2=KL1+R_{row}-1$ where R_{row}=PD.*.*TileNum*(row,1)

<u>Default case: 2-D data vector</u>

This section describes the use of repetition factors for those tiles whose default data is a 2-D array (think of a matrix of data values). For each "WFClp" data item above, the default data is a matrix vector such as PD.WFClp.Nsamps.*TileNum*(iray, ipi) , and the range of the indices is over all the tile's pulses and tile's rays when the flag is 0. Given the single return number for the tile, for each value of "ipi" and "iray" (that is, pulse-ray pair) there is, by default, one value for Nsamps. We illustrate the method with Nsamps, but it applies to RtnsCode and Tstart.

For there to be a lot of repeated values of Nsamps (as when many clips are the same length, many pulse-ray pairs within a fixed return value (that keeps the discussion within a single tile) would have to have the same Nsamps. For one pulse, many rays are likely to have similar waveforms. However, for very high PRFs and a lot of oversampling, one ray is likely to have even more instances of a given waveform over a large range of pulses. So if we 1-dimensionalize the basic 2-D array by defining an index, "kL" (read as "kay linear"), by $kL = (iray-1)\times n_{pulse} + ipi$ {0≤ipi≤ # of pulses in tile-1, 1≤iray≤n_{ray} = #of elements in focal plane array= length of the default matrix rows}, the "repetition ready" version of the array would be PD.WFClp.Nsamps.*TileNum*(kL,j). All the ray-pulse index pair information is moved to the first index and the second index, with a range of 3 values (e.g., 0:2 or 1:3) awaits population in the reduction by elimination of repeated rows. After deleting repeated rows, we fill in the column values (j indexed) with, (j=1) the ray index

[77] This is the value {e.g. Nsamps} for the requested logical row, KL. It is obtained from the array that has been reduced via repeat factors.

that the Nsamps on this row belongs to[78], (j=2) the repeat factor [79]for that ray-Nsamps value, and (j=3) a value for Nsamps. After elimination of repeats, the actual range of "ckL[80]" would be the number of separate repeat sequences in the tile, which is given by PD.WFClp.Nsamps.*TileNum*.Flg . Regarding tracking repeats, the j=2 entry is the number of times the value on that row occurs consecutively in a single running streak. If the value appears again after a different intervening value appears in a row, a new actual row in the tile has the repetition value for that later logical row sequence as a later separate entry in the actual tile array.

To find the value of Nsamps for a given ipi and iray (i.e. logical values) within a tile that uses repetition factors

1. ***Treat the default array as 1 dimensional by considering it to be a column of rows[81] with index*** kL ***derived from pulse and ray indices as given above.***
2. ***Formally define the repetition factor array's rows to have 3 elements: first is an identifier for the ray, second is a repetition factor, third is the value of Nsamps***
   R_k = PD.WFClp.Nsamps.*TileNum*(ckL,2),
   Value= PD.WFClp.Nsamps.*TileNum*(ckL,3).
   The range of ckL is over the number of actual rows, given by PD.IntTileMap.Nsamps.*TileNum*.Flg
3. ***We have reduced the problem to the previous 1-d treatment so-***
   We can ask for the value, say of Nsamps, for a ray and pulse to be extracted from the version of the array stored with repeat factors. Ask for the value for the logical row, kL(iray, ip) per above formula.

[78] We need to save this for unambiguous reconstruction since a single ray could contribute wave form clips with different (repeated) values of Nsamps in different pulse runs.

[79] That is, the number of consecutive pulses having the same Nsamps for the a given running streak of Nsamps for a ray.

[80] We introduce a new notation, "ckL" for an index that runs over the list of repeat sequences where a single ray has consecutive pulses with the same Nsamps in the wave clips of the return number of the tile. See below for "ckL" appearance as an index in the "columnized" array.

[81] A row in the linearized (we also can think of this as "columnized") array at this point has the value of Nsamps for the kL^{th} row major pairing of ray and pulse index value.

Actual row value (ckL) from logical row and column (iray,ipi)[82] :

```
ckL=1,  sum=0
Do while (iray>PD.*.TileNum(ckL,1))
     ckL=ckL+1
End do
We have jumped to the first row of the compressed array for the desired ray.
Now, get to the needed pulse range.
Do while (ipi>sum)
      sum= sum+ PD.*.TileNum(ckL,2)
      ckL=ckL+1
End do
Value= PD.*.TileNum(ckL-1,3)  will be the value {e.g. of Nsamps} for iray
                              and ipi obtained from the compressed version
                              of the data array.
```

This internal tile mapping mechanism might be considered a cross over into formatting and not truly transport independent conceptual data. However, to be truly transport independent and not presume a mechanism exists to hide the reduction of tile data volume while exposing only the logical data content means that the metadata needs to document the effect of sorting the data and populating the data arrays. The conceptual data model cannot demand that all data be zipped, and even it did, unzipping would cause the receiving application to have to use much more than the necessary memory space. In the world of parallel processing, the impact of large redundant data structures required for compliance with a data definition set could be devastating. Thus we explicitly define the metadata for internal tile mapping.

10.3. Generalization of VRP data

10.3.1 Fast time data—Waveform capture

The basic idea behind the CMMD treatment of lidar "wave form" data is that the received "fast time" photocurrent vectors are storable, mapped and documented at L-1 and L-2 . The data is stored in waveform clips whose sizes in terms of fast time samples are given as metadata. The wave form clips, which while typically relatively short in fast time compared to the full "range gate" or recording window for enterprise level-2 data, can be as long as the full recording window, as they all must be for enterprise level-1 data.

[82] This is how to get the value from the compressed version of the array when we know the ray and pulse for which we seek the value.

The CMMD strategy is to use the parameter, LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) to carry wave form data in both enterprise level-1 and level-2 data sets. Per the discussion in 10.1.1.3, this parameter has been defined so as to accommodate both Geiger Mode and Linear mode family[83] data. The wide band recorded lidar received data in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) is supported by a number of metadata items in SC.WF.*, which not only supply data about the transmitted pulse waveform [the waveform data for which are in SC.WF.TxProfile.Dat(il,j)], but also the ray directions for which the elements in the receiving detector arrays are sensitive as well as calibration and uncertainty metadata.

A waveform is uniquely identified by its pulse and ray identifier (iray). The ray identifier, in turn, combines the detector element identifier with the array on which the waveform was detected. Since the array and the pulse determine the channel[84] (which is specified by the choice of laser or pulse and array), we do not need to add the channel to the waveform for purposes of unique specification, but rather we can link the waveform to the channel that it belongs to through ES.WFClp.ChNum.*TileNum*(ipi, iray) as an exploitation support metadata item.

Because it is not generally supported by physics to associate an extended wave form with a single and unique XYZ position, the CMMD does not choose to use some time internal to the waveform to compute a reference XYZ position. However, since practicality demands that we be able to locate a waveform on the basis of an XYZ in which we are interested, the initial time in each waveform clip can be used to compute an XYZ location and stored in ES.WFClp.Locs.*TileNum*(ipi,iray,j). This is most reliably an accurate position only for a first return, and the calculation for each waveform clip for a given pulse-ray is computed as if it was a first return.

At level-2, lidar returns can be listed. At L-1, the nature of returns is not yet revealed and at L-3 and above, waveforms are not conveyed. In this CMMD, for a level-2 data set that contains both waveforms and points, the return counting scheme integrates the counting of returns of waveforms and points. Each waveform clip represents a single return and is counted as an independent return for a "pulse-ray". A single pulse-ray may yield both point returns and waveform clip returns. The sequence of returns for a pulse-ray-channel is interms of fast time, with the lower fast times corresponding to lower return numbers.

Note regarding waveform oriented tiles:

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile". *TileNum* is used to make logical associations of elements in 'table rows" where the entries in the "table columns" for any single row may include disparate scalars

[83] Pulse compression waveforms are in the linear mode family.

[84] SC.Channlzn.Descr(j,k) gives the description parameters for channels in terms of phenomenology. However, only a single laser- array paring is possible on a single pulse. If an array were exposed to returns from two lasers simultaneously, the data would be mixed and not represent separate channels of information. So the array and pulse determine what photons in spectral and polarization space are causing the recorded return data.

such as starting times and pulse numbers or vector quantities such as the fast time waveform records themselves.

Use of *TileNum* for wave form clips enterprise L-2:

Waveforms in enterprise L-2 data are generally subsets ("clips") of the full receive window record. The length of the subset in fast time is a parameter, and it is permissible that, on occasion, the waveform could cover the entire receive window. But the intent of L-2 data is to focus on physical echoes or "returns" from relatively small extent regions of the remotely sensed spatial volume, and consequently these generally will span a small fraction of the receive window duration. Typically, much of the return in a full waveform is at the noise or zero recording level. The waveform clips will be limited to those regions of interesting response. Multiple clips from a single pulse-ray may be stored as separate data vectors and these are characteristically associated with specific returns of interest. That is, at L-2, **waveform tiles are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "1R*nnnnnn*" needs to be unique. The integer, j_tile, can't be simply a transcription of the *TileNum*, but it can still be a unique numeric identifier for each value of *TileNum* used in a data set. PD.TileMap.TileNum(j_tile) will have the tile names listed in consecutive order and the j_tileth character string on the list can look like "*N*R*nnnnnn*" without breaking any reference correspondences.

Wave form clips are stored in dedicated tiles—no mixing of points and waveforms within the same vertical tile variable is permitted. The data items that are available to carry, notify about, and describe waveform data in this CMMD are:

MD.Objective.Abstract	SC.Swath.RateFactor(ia)
ES.WFClp.ChNum.*TileNum*(ipi, iray)	SC.Swath.TimeStep
ES.WFClp.Locs.*TileNum*(ipi,iray, j)	SC.WF.RayDesc.ArryGlbl(iray)
ES.WFClp.RtnsCode.*TileNum*(ip,iray, j)	SC.WF.RayDesc.ElmtGlbl(iray)
PD.HiIndx.AdjacentW(j)	SC.WF.RayDesc.Arry.*TileNum*(iray)
PD.LoIndx.AdjacentW(j)	SC.WF.RayDesc.Elmt.*TileNum*(iray)
PD.HiIndx.WF(j)	SC.WF.Rx.Nsamps(ich)
PD.LoIndx.WF(j)	SC.WF.Rx.IntnsyUncty(ich)
PD.HiIndx.WFClpPulseNum.*TileNum*	SC.WF.TxProfile.RF_Slope(il)
PD.LoIndx.WFClpPulseNum.*TileNum*	SC.WF.TxProfile.RF_Start(il)
PD.TileMap.AdjacencyWFs(n_tile)	SC.WF.TxProfile.RF_PhaseStart(il)
PD.TileMap.WF_Returns(ich,j)	SC.WF.TxProfile.Dat(il,j)
PD.WFClp.Nsamps.*TileNum*(ip,iray,j)	SC.WF.TxProfile.Flags(j)
PD.WFClp.PulseNum.*TileNum*(ipi)	SC.WF.TxProfile.Nsamps(il)
PD.Wordsize.RxDatVec	SC.WF.TxProfile.Peak(il)
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)	SC.WF.VQ_Decode(il, icode)

SC.ComprssM.Parms	LD.WFClp.RxDatVec.*TileNum*(ip,iray, it)
SC.ComprssM.Flags	LD.WFClp.Tstart.*TileNum*(ip,iray,j)

Some of these items are for pulse compression waveforms and would not be used for simpler linear mode systems.

10.4. User Defined data

This CMMD provides a framework for documenting data in a file that is not defined in a currently published release of the CMMD or any of its enterprise level focus documents. To provide this framework, we exploit the computable name capability of the DAL API to permit the documentation of 3 data item groups in the CMMD, but allowing many data items to be user defined in any particular data set, possibly several data items per group. The groups are defined below in two wide band categories and one flexible narrow band category. A user may define 0, 1 or more wide band items

at the point level via UD.UserDefined.PtDat.P*n*.*TileNum*(ipt)

at the pulse level via UD.UserDefined.PulseDat.P*n*.*TileNum*(ip), and

0,1 or more narrow band items

at the tile level via PF.UserDefined.NB_Dat.P*n*.*TileNum*(j).

Note that each group has a sub field "P*n*" in its data tag. The fixed character, "P" stands for "parameter". The computed string, *n*, can be given a unique value for each distinct parameter in the group. For example, if a user introduces 2 new point oriented parameters, the names could be as simple as "P1" and "P2", or as elaborate as "PSV1" and "PCPi". For practical purposes, the evaluated string, *n*, should be limited to 3 or less characters. If *n* were limited to the 10 numeric characters, the three groups provide names for 30 separate user defined quantities.

Note further that a *TileNum* sub-field is incorporated in the tags for each group. This is necessary for the wideband data to be retrievable by tile. But providing the *TileNum* field for the narrowband data permits a tile specific instantiation for single narrow band data items.

Having provided a generic framework for naming user defined data items, we have to offset the fact that these items have no definitions provided in the CMMD or enterprise level specific data definition documents. This is done via the PD.UserDefined.NB_Def.P*n*, PD.UserDefined.PtDef.P*n*, and PD.UserDefined.PulseDef.P*n* entries in the CMMD which provide text strings carrying the definitions for each user defined parameter as uniquely identified within each group by the string "P*n*". These definition strings are governed by PD.Wordsize.* parameters to support flexible levels of detail for the "in-file" user definitions of the non-standard parameters.

<u>Guidelines for populating User definition text strings</u>

Users should strive to provide definitions that meet the objectives of being unambiguous, explicit and supporting interoperability via semantic completeness, the same objectives as the metadata dictionary in this CMMD. In fact the "final" text definition of a

user defined field may ultimately become a metadata dictionary entry without change. The data type, is afforded a separate code in the parameters PD.UserDefined.NB_Type.P*n*, PD.UserDefined.PtType.P*n*, and PD.UserDefined.PulseType.P*n* for float/precision or [U]Int*N* form and presents no challenges to clear exposition. But to make the definition text more than a wink and nod between parties that have independent knowledge of the meaning of the user defined parameters, we provide the following guidelines.

- The definition should be divided in sections addressing a sequence of detail levels from top (first) level to lowest (4th)level
- At the top level, the entity type of the item should be specified as either 1)a measurement, 2)a quantity derived from measurement, 3) a quantity input by the user, 4) a quantity derived from inputs by the user, or 5) a code.
- At the second level, the dimensions of a measured quantity should be given. Dimensions are the measurable property, principally based on instrument phenomenology being reported, not the units of measure. Reportable dimensions include "dimensionless", and basic dimensions such as "distance", "time", "mass', "charge' and powers of such basic dimensions (e.g. distance^3 for a volume). For convenience, other such dimensions of long standing physics convention that could be expressed in terms of the fundamental dimensions can be reported (such as force, current, temperature, etc). Some common dimensionless ratios can be reported as if dimensioned in the strict sense. This includes "angle", "decibels", "Mach number"[85] etc.
- At the third level, the underlying entity whose type was identified at the top should be given. This is generally the name of a measurable, derived, or codable object such as "aperture radiance", " slope variance", or "method".
- At the fourth level, the units of measure and formulation should be provided. "Formulation" describes how measurements or derivations are done. For example, if the user defined field is used to convey an experimental slope variance, the string, "slope variance" might be used as a third level descriptor, but without the formulation one might not be able to determine whether the "slope variance" is the second derivative of z with respect to x and y, (i.e. $d^2z/dxdy$) or [1/N]Σ SQRT{[dz/dx]^2 +[dz/dy]^2}. This level is also where the meaning of any relevant codes is provided.
- Any unstated convention should yield to the defaults of this CMMD document. For example, if not explicitly stated, all angles are presumed to be measured counter-clockwise as viewed along the axis of rotation looking toward the

[85] So we declare the underlying dimensional quantity here {angle—though angles are dimensionless, power, speed(length/time)} and give the actual units, say, {degrees, dB and Mach Number} at the fourth level, below.

coordinate origin. Users should also avoid capricious violation of the conventions of this CMMD, such as defining an angle as measured clockwise.

20. APPENDIX: lidar Data Dictionary-- Detailed Data Item Definitions

20.1 MD.* True Metadata

MD.Aggregate.*

MD.Aggregate.* parameters describe associated data sets, to include parent and Fellow data sets to indicate heritage and complementary data sets derived from common heritage.

20.1.1 MD.Aggregate.Parent.CollctnDate(j) [L2,3,4,5]

Brief Definition	MD.Aggregate.Parent.CollctnDate(j) provides the date(s) of collection of the raw source data for the parent data set(s) of the current data set.
indices	Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents , that is over the number of parent data sets. For level 3 or lower data sets, only one parent data set exists.
Units/representation	Days, UInt16
Treatment/usage	Parent data sets are the data sets from which the current data set is directly derived. Since enterprise level-2, level-3, level-4, and level-5 are all <u>processed</u> data (not collected data), to determine when the raw data at the start of the processing chain for a given product at a level above L-1 was collected, one looks to the collection date of the parent of the current data set. That parent collection date will have been inherited, in turn, from its parent(s). The chain is initialized by populating the chain's first L-2 data set's value of MD.Aggregate.Parent.CollctnDate(j) with the value of MD.Collect.CollctnDate in the L-1 data set at the start of the chain, which is the parent of the first L-2 in the chain. In the event that the processing architecture does not use an enterprise L-1 data set at the start of its processing chains, the earliest product in the chain will have been populated by the level-0 date of collection, which is where the L-1 MD.Collect.CollctnDate is, in the end, populated from. MD.Aggregate.Parent.CollctnDate(j) is ultimately inherited from

MD.Collect.CollctnDate of the enterprise level-1 source data for the jth parent data set for lidar sources.

The inheritance of collection date data is illustrated in the case below:

Product in chain	**Level**	**Tag for collection date**
1	1	MD.Collect.CollctnDate ↓
2	2	MD.Aggregate.Parent.CollctnDate(1) ↓
3	3	MD.Aggregate.Parent.CollctnDate(1) ↓

For L-4, the metadata tag names acting as the sources for non-lidar sensors are defined in documents outside of this lidar CMMD[86].
Also, for level-4 products, there will be multiple chains like the one above feeding in. In that case, the range of j will be 2 values or greater.

MD.Aggregate.Parent.CollctnDate(j) follows the CMMD pattern of using the Truncated Julian Day (TJD) for expressing a date to year-month-day granularity. MD.Aggregate.Parent.CollctnDate(j) is the number of days elapsed between midnight on May 24, 1968 and the date of the start of the collection operation for the parent data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of

[86] Ideally, the NSG will adopt a common metadata scheme such that other sources will also use the metadata names MD.Collect.CollctnDate and MD.Aggregate.Parent.CollctnDate(j)

[87] **Day** normally refers to the *integer* day while **Date** (also Date-time) is commonly interpreted to include a *fractional* part that may be expressed in hours, minutes and seconds by converting the deciml fraction with the value 86400 sec/day. We use TJD as an abbreviation for Truncated Julian Day. Note that other users may intend the abbreviation, "TJD", to refer to the Date.

CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

It is essential that the ordering of each Parent.* data item in index j be the same as is used in MD.Aggregate.Parent.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also MD.Collect.StartTime MD.Collect.CollctnDate

20.1.2 MD.Aggregate.Parent.CollctnOpID(j) [L2,3,4,5]

Brief Definition MD.Aggregate.Parent.CollctnOpID(j) provides the identifier(s) for the collection operation(s) for the parent data set(s) of the current data set.

indices Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents , that is over the number of parent data sets. For level 3 or lower data sets, only one parent data set exists.

Units/representation Dimensionless, BCS Character (40)

Treatment/usage MD.Collect.CollctnOpID(j) gives the collection operation ID for each lidar collection operation used to create the current data set. For each value of j, this is a character string assigned by the mission control system for the collector sensor/platform to act as a tag for each single operational event that collected data represented in the data set. This will generally be unique within a day and may include encodings for sorties, departures, comms windows, etc. unique to the command and control system for the sensor. This metadata item is purely a bucket for an identifier and places no constraints on the symbols or structure beyond the limit of 40 characters and post padding with BCS blanks. Any interpretation of the content of this string is outside the control of the lidar CMMD.
It is essential that the ordering of each Parent.* data item in index j be the same as is used in MD.Aggregate.Parent.DS_Name(j) to consistently attribute information to each specific data set.

Parent data sets are the data sets from which the current data set is directly derived.
Since enterprise level-2, level-3, level-4, and level-5 are all processed data (not collected data), to determine the collection operation identifier for the raw data at the start of the processing chain for a given product at a level above L-1, one looks to the collection OpID of the parent of the current data

set. That parent OpID will have been inherited, in turn, from its parent(s). The chain is initialized by populating the chain's first L-2 data set's value of MD.Aggregate.Parent.CollctnOpID(j) with the value of MD.Collect.CollctnOpID in the L-1 data set at the start of the chain, which is the parent of the first L-2 in the chain. In the event that the processing architecture does not use an enterprise L-1 data set at the start of its processing chains, the earliest product in the chain will have been populated by the level-0 collection operation identifier, which is where the L-1 MD.Collect.CollctnOpID is, in the end, populated from.

MD.Aggregate.Parent.CollctnOpID(j) is ultimately inherited from MD.Collect.CollctnOpID of the enterprise level-1 source data for the jth parent data set for lidar sources.

The inheritance of collection date data is illustrated in the case below:

Product in chain	**Level**	**Tag for collection date**
1	1	MD.Collect.CollctnOpID ↓
2	2	MD.Aggregate.Parent.CollctnOpID(1) ↓
3	3	MD.Aggregate.Parent.CollctnOpID(1) ↓

For L-4, the metadata tag names acting as the sources for non-lidar sensors are defined in documents outside of this lidar CMMD[86].
Also, for level-4 products, there will be multiple chains like the one above feeding in. In that case, the range of j will be 2 values or greater.

Governing quality metrics

See Also MD.Collect.CollctnOpID

20.1.3 MD.Aggregate.Parent.DS_Name(j) [L2,3,4,5]

Brief Definition MD.Aggregate.Parent.DS_Name(j) identifies the data set(s) from which the current data set is directly derived.

indices Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of Parent data sets. For enterprise level 2, 3 data sets, only one parent data set exists.

Units/representation Dimensionless, BCS 60 characters

Treatment/usage MD.Aggregate.Parent.DS_Name(j) is inherited as the values of MD.Meta.DS_Name from each lidar parent data set. Refer to

MD.Meta.DS_Name for data set name coding and format. The metadata tag names acting as the sources for non-lidar sensors are defined in documents outside of this lidar CMMD.
It is essential that the ordering of each Parent.* data item in index j be the same as is used in MD.Aggregate.Parent.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also MD.Meta.DS_Name

20.1.4 MD.Aggregate.Parent.Duration(j) [L2,3,4,5]

Brief Definition MD.Aggregate.Parent.Duration(j) is the difference between the first collected data element's collection time and the last collected data element's collection time for the jth parent data set.

indices Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of Parent data sets. For level 3 or lower data sets, only one parent data set exists. One can expect the range of j to be two or more for enterprise level-4 data sets.

Units/representation Seconds, single Float

Treatment/usage MD.Aggregate.Parent.Duration(j) gives the length in seconds of the collection (slow) time interval represented in the jth parent of the current data set. Level-2 and level-3 data sets have only a single parent, so this is the information on the retainable interval of the underlying collection consistent with the data for the enterprise level-1 data set(s) at the root of the PED chain for the current data set. Using a duration avoids having to provide both a date and time for the latest point or waveform in a lidar data set.
Together with MD.Aggregate.Parent.CollctnDate(j) and MD.Aggregate.Parent.StartTime(j), this determines the interval of temporal coverage for the parent data set. This information is for discovery purposes and does not necessarily account for the impacts of processing on the temporal span represented by individual data points in the data set. So, this would properly be based on times associated with the input to the current data set, and not necessarily the time tags for the points output in this data set.

Governing quality metrics

See Also MD.Collect.CollctnDate MD.Coverage.Temporal.Duration

20.1.5 MD.Aggregate.Parent.StartTime(j) [L2,3,4,5]

Brief Definition MD.Aggregate.Parent.StartTime(j) indicates the times of collection start for the parent data sets of the current data set.

indices Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over

	the number of Parent data sets.
Units/representation	Seconds, single Float
Treatment/usage	Parent data sets are the data sets from which the current data set is directly derived. Since enterprise level-2, level-3, level-4, and level-5 are all processed data (not collected data), to determine when the raw data at the start of the processing chain for a given product at a level above L-1 was collected, one looks to the collection time of the parent of the current data set. That parent collection time will have been inherited, in turn, from its parent(s). The chain is initialized by populating the chain's first L-2 data set's value of MD.Aggregate.Parent.StartTime(j) with the value of MD.Collect.StartTime in the L-1 data set at the start of the chain, which is the parent of the first L-2 in the chain. In the event that the processing architecture does not use an enterprise L-1 data set at the start of its processing chains, the earliest product in the chain will have been populated by the level-0 date of collection, which is where the L-1 MD.Collect.StartTime is, in the end, populated from. MD.Aggregate.Parent.StartTime(j) is ultimately inherited from MD.Collect.StartTime of the enterprise level-1 source data for the jth parent data set for lidar sources. The range of index, j, in MD.Aggregate.Parent.StartTime(j) is only 1 for enterprise level-2,3 data sets, since there is only one direct precursor for these levels. Data sets at level 4 (and a level-5 derived from a level-4) may be based on multiple collections and so there will likely be multiple start times for the parents of such data sets, requiring a range for index, j, greater than one. MD.Aggregate.Parent.StartTime(j) for each value of j is the start time of the collection operation for parent (j) of the current data set. This represents the number of SI seconds past midnight of the TJD[88] when the sensor began collecting data for the jth parent of the current data set. An implementation of a GUI can display MD.Aggregate.Parent.StartTime as HHMMSS.sssssss, but storing the metadata in seconds units permits easy numerical manipulation.

[88] An operational guard band excluding start times within one minute around midnight will prevent discrepancies between UTC and TJD from creating anomalously large/small start times for early/late in the day operations for those using UTC references.

Single float precision will permit start times to be stated to a precision of at least 0.01 seconds on any day. It is recommended that mission control systems start collections on intervals of 0.01 second granularity to avoid ambiguous ordering of start times based on single precsion float data.

Together with MD.Aggregate.Parent.CollctnDate(j), this determines the day and time of collection start for a parent data set of a particular (lidar) product.

It is essential that the ordering of each Parent.* data item in index j be the same as is used in MD.Aggregate.Parent.DS_Name(j) to consistently attribute information to each specific data set.

Note that the start of collection time for the parent data set(s) is not necessarily the earliest time for wideband data in the current data set. For that data need, see MD.Coverage.Temporal.StartTime.

Governing quality metrics

See Also MD.Coverage.Temporal.StartTime PD.HiIndx.Parents MD.Collect.StartTime

20.1.6 MD.Aggregate.Fellow.CollctnDate(j) [L1,2,3]

Brief Definition MD.Aggregate.Fellow.CollctnDate(j) is the date of collection of data in the fellow data products.

indices Index, j, ranges from PD.LoIndx.Fellow to PD.HiIndx.Fellow, that is over the number of cited fellow data sets.

Units/representation Days, UInt16

Treatment/usage Fellow data sets are data sets that have been formed from collections with the intent that they be co-exploited and/or co-processed. Reference and match sets, geometrically diverse collections of single scene regions are common types of fellow data sets. The separate data sets satisfying an area collection requirement that have mutual coverage or boundaries are fellows as well. That is, a group of data sets collected with the intent of eventual mosaicking contains sub-groups that are fellow data sets identifiable by their adjoining coverage. “Fellows” also include data sets with at least one common precursor, as they are natural candidates for co-exploitation. When the collection and processing plan anticipates the creation of fellow data sets, it is an easy matter to populate metadata in each of the fellows at the time of their generation. This is the case where we are most likely to see MD.Aggregate.Fellow.CollctnDate(j) populated. The MD.Aggregate.Fellow.* data is cited in a file as a convenience.

This CMMD provides support for a list of fellow data sets of length determined by PD.Lo/HiIndx.Fellows. The CMMD has no control over the completeness of this list in any data set or any implementation.

MD.Aggregate.Fellow.CollctnDate(j) is the day of collection of data for the jth fellow data product, expressed as Truncated Julian Day.

For discovery and retrieval, collection date is expressed as Truncated Julian Day (TJD). TJD, introduced by NASA in 1968 for machine implementation convienience, is the number of days elapsed between midnight on May 24, 1968 and the date of the start of the collection operation for the referenced data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. The Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, see e.g. http://www.batse.msfc.nasa.gov/tools/date_convert/ or http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also PD.HiIndx.Fellows

20.1.7 MD.Aggregate.Fellow.CollctnOpID(j) [L1,2,3,4,5]

Brief Definition MD.Aggregate.Fellow.CollctnOpID(j) provides the identifiers(s) for the collection operations for the fellow data set(s) of the current data set.

indices Index, j, ranges from PD.LoIndx.Fellows to PD.HiIndx.Fellows, that is over

	the collections contributing to the cited fellow data set list.
Units/representation	Dimensionless, BCS Character (40)
Treatment/usage	MD.Collect.CollctnOpID(j) gives the collection operation ID for the collection operation used to create the fellow data set. For each value of j, this is a character string assigned by the mission control system of the sensor that collected for the cited fellow data set. The Operation IDs for the collector sensor/platform act as tags for each single operational event that collected data. This will generally be unique within a day and may include encodings for sorties, departures, comms windows, etc. unique to the command and control system for the sensor. This metadata item is purely a bucket for an identifier and places no constraints on the symbols or structure beyond the limit of 40 characters and post padding with BCS blanks. Any interpretation of the content of this string is outside the control of the lidar CMMD. It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.
Governing quality metrics	
See Also	

20.1.8 MD.Aggregate.Fellow.DS_Name(j) [L1,2,3,4,5]

Brief Definition	MD.Aggregate.Fellow.DS_Name(j) identifies fellow data sets of the current data set.
indices	Index, j, ranges from PD.LoIndx.Fellow to PD.HiIndx.Fellow, that is over the number of cited fellow data sets.
Units/representation	Dimensionless, BCS character(60)
Treatment/usage	MD.Aggregate.Fellow.DS_Name(j) is a character string containing the data set name for the jth cited fellow data set. Refer to MD.Meta.DS_Name for data set name coding and format. It is essential that the ordering of each MD.Aggregate.Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.
Governing quality metrics	
See Also	MD.Meta.DS_Name

20.1.9 MD.Aggregate.Fellow.PlatformID(j) [L1,2,3,4,5]

Brief Definition	MD.Aggregate.Fellow.PlatformID(j) identifies the platform(s) from which data for the cited fellow data sets have been collected.
indices	Index, j, ranges from PD.LoIndx.Fellows to PD.HiIndx.Fellows, that is over

the number of fellow data sets.

Units/representation: Dimensionless, BCS character(6)

Treatment/usage: MD.Aggregate.Fellow.PlatformID(j) provides the values of platform identifiers for fellow data sets. For lidar data set fellows, this would be from MD.Platform.ID(j) for the fellow data set. The metadata tag names acting as the platform identifiers for non-lidar sensors are defined in documents outside of this lidar CMMD.

It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also: MD.Platform.ID(j)

20.1.10 MD.Aggregate.Fellow.ProdType(j) [L1,2,3,4,5]

Brief Definition: MD.Aggregate.Fellow.ProdType(j) indicates the product level and stage of fellow data sets.

indices: Index, j, ranges from PD.LoIndx.Fellow to PD.HiIndx.Fellow, that is over the number of cited fellow data sets.

Units/representation: Dimensionless, UInt8

Treatment/usage: MD.Aggregate.Fellow.ProdType(j) gives the code for the character string in MD.ProcHist.CodeList that describes the product type of the jth fellow data set.

The product type character string contains a product level code and a (list of) product stage code(s).
The codes in MD.Aggregate.Fellow.ProdType(j) resolve to a character string such as “L2.S1bb.S2a.S2b1". Refer to MD.ProcHist.ProdType for details of product and processing stage sub-string codes and construction of entries in the code list table.

It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also: MD.ProcHist.ProdType MD.ProcHist.CodeList(j)

20.1.11 MD.Aggregate.Fellow.SensorID(j) [L1,2,3,4,5]

Brief Definition: MD.Aggregate.Fellow.SensorID(j) identifies the sensor ID(s) that collected data for the cited fellow data sets.

indices	Index, j, ranges from PD.LoIndx.Fellows to PD.HiIndx.Fellows, that is, over the number of fellow data sets.
Units/representation	Character String, BCS, 6 characters
Treatment/usage	MD.Aggregate.Fellow.SensorID(j) provides the values of sensor identifiers for cited fellow data sets. These identifiers should be registered to assure common referencing of data sources. For lidar data set fellows, MD.Aggregate.Fellow.SensorID(j) would have the value of MD.Sensor.ID(j) in the cited fellow data set. The metadata tag names acting as the sensor identifiers for non-lidar sensors are defined in documents outside of this lidar CMMD. It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.
Governing quality metrics	
See Also	MD.Sensor.ID(j)

20.1.12 MD.Aggregate.Fellow.SensorType(j) [L1,2,3,4,5]

Brief Definition	MD.Aggregate.Fellow.SensorType(j) indicates the type of sensor, e.g. LIDAR, EO color, Video MWIR, … used for fellow data set collection.
indices	Index, j, ranges from PD.LoIndx.Fellow to PD.HiIndx.Fellow, that is over the number of cited fellow data sets.
Units/representation	Dimensionless, BCS character(2)
Treatment/usage	MD.Aggregate.Fellow.SensorType(j) indicates the type(s) of sensor(s) that collected the data in fellow data sets. Code: 00— N/A for fellow data sets 1-- same LIDAR as current data set 2-- Context EO color 3-- Context Video color (MSI/HSI) 4-- Context EO pan vis 5-- Context EO Stokes 6-- Context EO SWIR 7-- Context Video SWIR 8-- Context EO MWIR 9-- Context Video MWIR 10-- SAR ka 11-SAR ku 12-SAR x

13-SAR c
14-SAR L
15-SAR p
16-- other LIDAR than current data set
17-- other EO color
18-- other Video color (MSI/HSI)
19-- other EO pan vis
20-- other EO Stokes
21-- other EO SWIR
22-- other Video SWIR
23-- other EO MWIR
24-- other Video MWIR

It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also MD.Sensor.lidarType

20.1.13 MD.Aggregate.Fellow.StartTime(j) [L1,2,3,4,5]

Brief Definition MD.Aggregate.Fellow.StartTime(j) indicates the times of collection start for the cited fellow data sets of the current data set.

indices Index, j, ranges from PD.LoIndx.Fellows to PD.HiIndx.Fellows, that is over the number of Fellow data sets.

Units/representation Seconds, single Float

Treatment/usage MD.Aggregate.Fellow.StartTime(j) for each value of j is the start time of the collection used to build the jth cited fellow data set. This represents the number of SI seconds past midnight of the TJD[89] when the sensor began collecting data for the jth fellow data set of the current data set.
An implementation of a GUI can display MD.Aggregate.Parent.StartTime as HHMMSS.sssssss, but storing the metadata in seconds units permits easy numerical manipulation.

Single float precision will permit start times to be stated to a precision of at least 0.01 seconds on any day. It is recommended that mission control systems start collections on intervals of 0.01 second granularity to avoid ambiguous ordering of start times based on single precsion float data.

[89] An operational guard band excluding start times within one minute around midnight will prevent discrepancies between UTC and TJD from creating anomalously large/small start times for early/late in the day operations for those using UTC references.

	Fellow data sets at level 4 can derive from multiple collections, but only the most recently collected parent's data start time is conveyed in this field. Together with MD.Aggregate.Fellow.CollctnDate(j), this determines the day and time of collection start for a fellow data set of the current lidar product. It is essential that the ordering of each Fellow.* data item in index j be the same as is used in MD.Aggregate.Fellow.DS_Name(j) to consistently attribute information to each specific data set.
Governing quality metrics	
See Also	

MD.Collect.*

MD.Collect.* parameters describe the items associated with characterizing the collection operation for raw data used to build the current data set and are principally for enterprise level-1. Only duration applies to enterprise level-2 data sets. The collection operation information at level-2,3,4, and 5 is carried in MD.Aggregate.Parent.*

20.1.14 MD.Collect.Duration [L1,2,3,5]

Brief Definition	MD.Collect.Duration gives the length of the temporal interval spanned by the collection operation from which the current data set has been formed.
indices	none
Units/representation	Seconds, float
Treatment/usage	This item supplants MD.Coverage.Temporal.Duration at enterprise level-1 only. The temporal duration for level-2 data sets and above is provided by MD.Coverage.Temporal.Duration. By continuing to inherit MD.Collect.Duration at higher product levels, the ability to advise the user of the original collection operation duration is maintained while also advising the user of the current product duration in MD.Coverage.Temporal.Duration and the direct parent temporal coverage in MD.Aggregate.Parent.Duration(j).
Governing quality metrics	
See Also	MD.Coverage.Temporal.Duration

MD.Coverage.*

MD.Coverage.* parameters describe the temporal span and geographic spatial region represented in the data set.

20.1.15 MD.Coverage.BE_NumList(j) [L1,2,3,4,5]

Brief Definition	MD.Coverage.BE_NumList(j) is the list of Basic Encyclopedia numbers covered by the current data set.
indices	Index j ranges from PD.LoIndx.Swath(6) to PD.HiIndx.Swath(6), that is over the number of BE numbers covered by the data set.
Units/representation	Dimensionless, BCS string, length 15
Treatment/usage	MD.Coverage.BE_NumList(j) supplies a list of Basic Encyclopedia numbers that apply to the data set. Each 15 character string consists of a basic 10 character BE number with an optional 5 character extension. This extension will be post padded blanks when not applicable.
Governing quality metrics	
See Also	PD.HiIndx.Swath(6)

20.1.16 MD.Coverage.Bounds.LatNorth [L1,2,3,4,5]

Brief Definition	MD.Coverage.Bounds.LatNorth contains the northern most latitude of the coverage area
indices	None
Units/representation	Degrees, single Float
Treatment/usage	If the product coverage is viewed as circumscribed by a geodetic "rectangle" with great circle constant longitude sides running north to south and minor circle constant latitude sides running west to east, then MD.Coverage.Bounds.LatNorth gives the north side of this rectangle. The value is nominal for discovery. The value in decimal degrees will be able to represent 3 meter intervals with latitudes given to seven significant figures.
Governing quality metrics	
See Also	MD.Coverage.Bounds.LatSouth MD.Coverage.Bounds.LonEast MD.Coverage.Bounds.LonWest

20.1.17 MD.Coverage.Bounds.LatSouth [L1,2,3,4,5]

Brief Definition	MD.Coverage.Bounds.LatSouth contains the southern most latitude of the coverage area
indices	None
Units/representation	Degrees, single Float
Treatment/usage	If the product coverage is viewed as circumscribed by a geodetic "rectangle" with great circle constant longitude sides running north to south and minor circle constant latitude sides running west to east, then MD.Coverage.Bounds.LatSouth gives the south side of this rectangle. The value is nominal for discovery. The value in decimal degrees will be able to represent 3 meter intervals with latitudes given to seven significant figures.

Governing quality metrics	
See Also	MD.Coverage.Bounds.LatNorth MD.Coverage.Bounds.LonEast MD.Coverage.Bounds.LonWest

20.1.18 MD.Coverage.Bounds.LonEast [L1,2,3,4,5]

Brief Definition	MD.Coverage.Bounds.LonEast contains the eastern most longitude of the coverage area
indices	None
Units/representation	Degrees, single Float
Treatment/usage	If the product coverage is viewed as circumscribed by a geodetic "rectangle" with great circle constant longitude sides running north to south and minor circle constant latitude sides running west to east, then MD.Coverage.Bounds.LonEast gives the east side of this rectangle. The value is nominal for discovery. The value in decimal degrees will be able to represent 3 meter intervals with longitudes given to seven significant figures.
Governing quality metrics	
See Also	MD.Coverage.Bounds.LatSouth MD.Coverage.Bounds.LatNorth MD.Coverage.Bounds.LonWest

20.1.19 MD.Coverage.Bounds.LonWest[L1,2,3,4,5]

Brief Definition	MD.Coverage.Bounds.LonWest contains the western most longitude of the coverage area.
indices	None
Units/representation	Degrees, single Float
Treatment/usage	If the product coverage is viewed as circumscribed by a geodetic "rectangle" with great circle constant longitude sides running north to south and minor circle constant latitude sides running west to east, then MD.Coverage.Bounds.LonWest gives the west side of this rectangle. The value is nominal for discovery. The value in decimal degrees will be able to represent 3 meter intervals with longitudes given to seven significant figures.
Governing quality metrics	
See Also	MD.Coverage.Bounds.LatSouth MD.Coverage.Bounds.LonEast MD.Coverage.Bounds.LatNorth

20.1.20 MD.Coverage.Bounds.ZoneList(j) [L1,2,3,4,5]

Brief	MD.Coverage.Bounds.ZoneList(j) is a list of the MGRS grid zone designators

Definition	defined as intersecting the current data set.
indices	Index j, ranges from PD.LoIndx.Zones to PD.HiIndx.Zones, that is over the number of zones defined to cover the current data set.
Units/representation	Dimensionless, BCS string, length given by PD.Wordsize.Zones
Treatment/usage	The ordering of the MGRS GZDs in this vector is the alphanumeric sort of the BCS zone designators. Each entry in MD.Coverage.Bounds.ZoneList(j) is a bucket for a character string carrying an MGRS grid zone designator. For convenience and clarity, the MGRS zone scheme is described below. MGRS grid zone designators (GZDs) reference zones of different areas and locations. An MGRS grid reference does not describe a point on the earth's surface, but rather describes a zone, a "rectangular"[90] area of size determined by the number of characters in the grid zone designator. The total number of characters defining the grid zone must be 3, 5, 7, 9, 11, 13, or 15 identifying the desired precision. All points within that rectangle share the same MGRS "coordinate" since the MGRS is not a coordinate system, but rather a zone referencing system and all the points lie in the same zone. When changing precision levels, it is important to truncate rather than round the easting and northing values to ensure the more precise polygon will remain within the boundaries of the less precise polygon. GZD designations and precision alternatives*: 04Q[91] .....precision level 6° × 8° (in most cases) 04QFJ .....precision level 100 km 04QFJ16 .....precision level 10 km 04QFJ1267 .....precision level 1 km 04QFJ123678 .....precision level 100 m 04QFJ12346789 .....precision level 10 m 04QFJ1234567890 .....precision level 1 m • The first two characters of an MGRS GZD represent the 6° wide UTM zone. • The third character is a letter indicating further partitioning into latitude bands. The letters C through X (excepting I & O) designate 8° MGRS bands of latitude from 80°S to 84°N (the northern most band is 12°). The letters A and B are used for Universal Polar Stereographic (UPS) designations in the

[90] MGRS is a variant of UTM. So it refers to a projection from the ellipsoid to a cylinder, which viewed as unrolled to a plane, is flat. The cells have length units in meters in this plane that measure the northing and easting. Each cell has the same easting width at each northing and the same northing width at each easting. Northing and easting within a cell is referenced to the SW corner.

[91] We choose to always use a length two zone number to represent the choices 1,2,3,…60. While some conventions permit using only one character for zones 1 through 9, we use the leading 0 for consistency.

southern polar region, and Y and Z for UPS designations in the northern polar region.

1. The fourth and fifth characters** (FJ in the illustration, above) are a pair of letters designating one of the 100,000-meter grid squares within the MGRS grid zones (or UPS area in the polar regions).
2. Following the above described leading characters, there follows an even number of numeric characters from the set {0,1,2,3,4,5,6,7,8,9}. The number of numeric characters can be up to 10, but since that addresses a zone of only 1 meter by 1 meter, it is an impractical choice for remotely sensed lidar point clouds. The first half of the numeric string represents the easting of the zone's south west corner and the second half represents the northing of the zone's south west corner (both measured from the SW corner of the 100km zone indicated in the first 5 characters). The first digit of each half gives the number of 10km steps, the second the number of 1km steps, the third the number of 100m steps, the fourth the number of 10m steps and the 5th the number of 1m steps from the south west corner of the 100,000m square denoted by the 4th and 5th letters of the zone designator. Thus, the longer the designator, the finer the sizing of the referenced zone.

It follows from the above rules for constructing an MGRS zone designator that if we have a 10km x 10km product aligned with the MGRS zones, we could reference it with a 7 character string. But we could also consider it to be 10 x 10=100 one km zones, each using 9 characters to designate the individual zone. For sufficiently dense point cloud data, the 1x1km blocks could be reasonable tile sizes. Clearly, the number of zones in a data set of a given horizontal extent depends on the size of the zones, which in turn is determined by the length of the string used to name the zones.

* Example using 04QFJ from wikipedia.

Governing quality metrics

See Also

PD.HiIndx.Zones PD.Wordsize.Zones
http://earth-info.nga.mil/GandG/coordsys/grids/referencesys.html
**https://www1.nga.mil/ProductsServices/GeodesyGeophysics/Coordinates/Related%20Documents/Military_Grid_Reference_System.pdf

20.1.21 MD.Coverage.Bounds.OuterZoneList(j) [L1,2,3,4,5]

Brief Definition

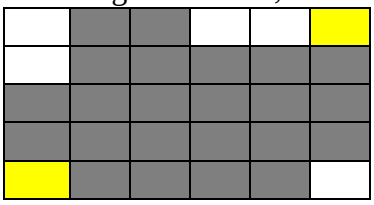
MD.Coverage.Bounds.OuterZoneList(j) lists the four spatial-extent-defining

MGRS GZDs.

indices
j=1 - Northmost zone
j=2 - Eastmost zone
j=3 - Southmost zone
j=4 - Westmost zone

Units/representation
MGRS grid zone designator (GZD), BCS string

Treatment/usage
This vector contains the extreme northern, southern, eastern, and western MGRS zones intersecting the current data set.
The entire data set could be contained within fewer than four zones, resulting in duplicate entries in this four element vector.
If it could be assured that the zones in a product always fill a rectangular region in the UTM projection of the product coverage, we would only need two diagonal corners to establish the zone extent of the data set. However, the CMMD is designed to permit products to be generated when the collection or processed data coverage does not fill a rectangle. In the following illustration,



the dark cells are the only zones populated with data in a sample data. The extent of the data requires 4 zones to be called out. Either populated zone in the top row and either populated zone in the left most column provide give the north and west bounds. Any choice of zone in the bottom row or right most column gives the south and east bounds, respectively. The application populating MD.Coverage.Bounds.OuterZoneList(j) is free to choose which zones to put into the outer zone list when there are multiple choices that act as bounding zones.
Only when all four corners are populated with data could we say that that the SW and NE zones define the extent of the data. The yellow corners act as the south and west most bounds and the north and east most bounds, thus giving all 4 bounds. But if the data is only populated where shaded, there is no single tile that is both north most and west most.

Governing quality metrics

See Also
PD.TileMap.Adjacency(ij)

20.1.22 MD.Coverage.Bounds.ZoneRanges(j) [L1,2,3,4,5]

Brief Definition
MD.Coverage.Bounds.ZoneRanges(j) contains the number of MGRS GZDs spanning the latitude and longitude extents of the current data set.

indices
Index j = 1,2

Units/representation Dimensionless, UInt8

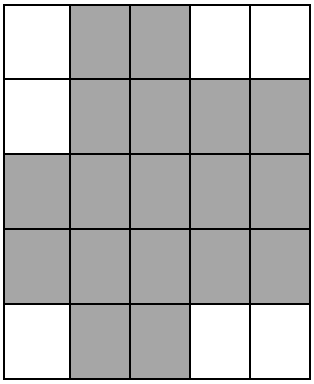
Treatment/usage This metadata item is to facilitate discovery and indicates the number of MGRS zones (at the grid zone size set by PD.Wordsize.Zones) that span the circumscribing rectangular region that includes the populated zones in the current data set.

MD.Coverage.Bounds.ZoneRanges(1) is the number of GZDs in latitude extent.

MD.Coverage.Bounds.ZoneRanges(2) is the number of GZDs in longitude extent.

When the data set is composed of a filled "rectangle" of MGRS grid zones, the circumscribing rectangular region dimensions are also the dimensions of the data coverage grid in MGRS zone units. In that case, *.ZoneRanges(1) × *.ZoneRanges(2) has the value of (PD.HiIndx.Zones- PD.LoIndx.Zones+1) See diagram below.

|←*Bounds.ZoneRanges(2) →| =5 zones



|←*Bounds.ZoneRanges(1) →| =5

This information is somewhat analogous to the tile mapping information, which also specifies the size of a grid, but rather than the zones in the present case, PD.TileMap.AdjacencyPts(n_tile) and PD.HiIndx.Adjacency work in units of tiles that have been sorted to represent horizontal geographic coverage. Making a tile match a zone is a good strategy.

Governing quality metrics

See Also MD.Coverage.Bounds.OuterZoneList PD.HiIndx.Adjacency PD.Wordsize.Zones PD.TileMap.AdjacencyPts(n_tile)

20.1.23 MD.Coverage.Bounds.Vertices(iv,j) [L1,2,3,4,5]

Brief Definition MD.Coverage.Bounds.Vertices(iv,j) is a vector of polygon vertices bounding the coverage area.

indices Index, iv, ranges from PD.LoIndx.Swath(4) to PD.HiIndx.Swath(4), that is,

	over the polygon vertices j=1-->Lat j=2-->Lon
Units/representation	Degrees, Float
Treatment/usage	MD.Coverage.Bounds.Vertices(iv,j) contains a counter-clockwise[92] oriented list of coverage area bounding vertices, beginning at the North East most vertex. The polygon defined by this list of vertices tightly circumscribes the lidar return locations from the nominal earth surface. Each vertex is the latitude, longitude pair giving the location of a point at the edge of the area covered by lidar measurements. A latitude, longitude pair does not change on a single ellipsoid normal, so these vertices are defined regardless of the surface forming the base of the lidar data in 3 dimensions. The polygon described by these vertices outlines the single swath of lidar product contained in the current data set. This polygon circumscribes the data set in latitude-longitude space on the reference[93] surface.
Governing quality metrics	
See Also	PD.HiIndx.Swath(4)

20.1.24 MD.Coverage.Bounds.Z(j) [L1,2,3,4,5]

Brief Definition	MD.Coverage.Bounds.Z(j) identifies the elevation bounds of the current data set.
indices	j=1 min Z, j=2 max Z, j=3 tasked minimum collectable
Units/representation	meters, Float
Treatment/usage	This parameter provides the minimum and maximum elevation above the EGM96 [or most current model as of the data set generation date] geoid of possible 3D point positions in the current data set. The EGM provides a consensus MSL datum that is consistent with Google Earth and GPS. This is a more physically meaningful datum than the ellipsoid fit to the geoid. This is nominal and for enterprise level-1 data reflects the range gating, not the as yet to be processed points. For point cloud data, this is Z extent of the Cartesian cube in which the processed lidar data could lie. It does not mean that the max Z point has been reported in this value. The highest Z value could be substantially less that what is indicated by MD.Coverage.Bounds.Z(j). If there were objects visible to the lidar with Z above MD.Coverage.Bounds.Z(2), they would have been excluded from the

[92] As if looking down on the earth surface .

[93] e.g. the WGS-84 ellipsoid or a planetary surface model for the region of lidar data in the data set.

data set, but objects visible to the lidar with Z between MD.Coverage.Bounds.Z(1) and MD.Coverage.Bounds.Z(2) should be in the data set unless they have been filtered out in the processing. MD.Coverage.Bounds.Z(3) is the minimum height coverage permitted for the horizontal coverage calculation. That is, when the horizontal extent of the product was computed, a height of MD.Coverage.Bounds.Z(3) meters above the base (given by MD.Coverage.Bounds.Z(1)) must have been within the sensor response range to permit claiming the horizontal coverage. This is to indicate whether absence of elevated returns near the boundaries of the product is merely a consequence of collection geometry or something more due to the scene structure. This data should come from tasking sources. A default value of 30 meters is to be assumed if this item is not populated. This is most informative in the case of off nadir collection where some ground coverage may have negligible vertical column above it and would be misleading if counted towards product coverage.

Governing quality metrics

See Also

20.1.25 MD.Coverage.Centroid[L1,2,3,4,5]

Brief Definition: MD.Coverage.Centroid(j) contains the 3D centroid of the current data set.

indices: j = 1 latitude, j = 2 longitude, j = 3 elevation

Units/representation: Latitude in degrees, longitude in degrees (decimal fractional parts), Z(elevation) in meters above reference geoid. Floats

Treatment/usage: This nominal value of the geometric center of the sensor returns represented in the data set is intended for discovery and data selection. The value is computed without regard to the intensity of sensor return and is representative of the target scene location only, not the sensor response to the scene. It is intended to be the location in the earth frame of the scene center for the product, not any particular point return in the product.

Governing quality metrics

See Also: ES.Origin.GeodtcEpoch

20.1.26 MD.Coverage.Cntry.Cd2(j) [L1,2,3,4,5]

Brief Definition: MD.Coverage.Cntry.Cd2(j) lists countries touched by lidar data coverage in the data set.

indices: Index, j, ranges from PD.LoIndx.Swath(5) to PD.HiIndx.Swath(5), that is over the number of countries in the data set

Units/representation: Dimensionless, BCS string of length 2

Treatment/usage: MD.Coverage.Cntry.Cd2(j) carries one of the ISO 3166 alpha-2 country codes in each slot indexed by j. Either MD.Coverage.Cntry.Cd2(j),

MD.Coverage.Cntry.Cd3(j) or both may be populated to carry country code information on a product. The logical representation of country code is unique to the specific country code tag. For code values, see http://www.iso.org/iso/country_codes/iso_3166_code_lists/iso-3166-1_decoding_table.htm

Governing quality metrics

See Also PD.HiIndx.Swath(5)

20.1.27 MD.Coverage.Cntry.Cd3(j) [L1,2,3,4,5]

Brief Definition MD.Coverage.Cntry.Cd3(j) lists countries touched lidar data coverage in the data set.

indices Index, j, ranges from PD.LoIndx.Swath(5) to PD.HiIndx.Swath(5), that is over the number of countries in the data set

Units/representation Dimensionless, BCS string of length 3

Treatment/usage MD.Coverage.Cntry.Cd3(j) carries one of the ISO 3166 alpha-3 country codes in each slot indexed by j. Either MD.Coverage.Cntry.Cd2(j), MD.Coverage.Cntry.Cd3(j) or both may be populated to carry country code information on a product. The logical representation of country code is unique to the specific country code tag. Per the ISO web site (http://www.iso.org/iso/store.htm), "The alpha-3 code is not made available free of charge. You can buy the International Standard ISO 3166-1 from our ISO Store. It contains the three-letter code."

Governing quality metrics

See Also PD.HiIndx.Swath(5)

20.1.28 MD.Coverage.Temporal.Duration [L2,3,4,5]

Brief Definition MD.Coverage.Temporal.Duration gives the length of the temporal coverage interval of the present data set.

indices none

Units/representation Seconds, Float

Treatment/usage This item is for enterprise levels 2,3,4,5. For L-1, use MD.Collect.Duration in lieu of MD.Coverage.Temporal.Duration. Inheriting MD.Collect.Duration while populating MD.Coverage.Temporal.Duration at levels 2,3,4,5 enables the original collection operation duration to remain available for users of downstream products that may have had their temporal coverage truncated.

Data set durations for level-2 and level-3 data sets are less than or equal to the parent data set durations [see MD.Aggregate.Parent.Duration(j)]. MD.Coverage.Temporal.Duration gives the length in seconds of that portion

of the collection (slow) time interval represented in the current data set. Using a duration avoids having to provide both a date and time for the latest point or waveform in a lidar data set. Together with MD.Coverage.Temporal.StartDate(j) and MD.Coverage.Temporal.StartTime(j), this determines the interval of temporal coverage for the data set.

This information is for discovery purposes and does not necessarily account for the detailed impacts of processing on the temporal span represented by individual data points in the data set. So, this would properly be based on times associated with the input to the current data set, and not necessarily the time tags for the points output in this data set.

Governing quality metrics

See Also MD.Aggregate.Parent.Duration(j) MD.Coverage.Temporal.StartDate(j) MD.Collect.Duration

20.1.29 MD.Coverage.Temporal.StartDate [L2,3,4,5]

Brief Definition MD.Coverage.Temporal.StartDate gives the day of collection for the earliest data in the current data set.

indices none

Units/representation Days, UInt16

Treatment/usage Since enterprise level-2, level-3, level-4, and level-5 are all processed data (not collected data), to determine when the raw data at the start of the processing chain for a given product at a level above L-1 was collected, one looks to the collection date of the parent of the current data set. That parent's collection date will have been inherited, in turn, from its parent(s). The chain is initialized by populating the chain's first L-2 data set's value of MD.Aggregate.Parent.CollctnDate(j) with the value of MD.Collect.CollctnDate in the L-1 data set at the start of the chain, which is the parent of the first L-2 in the chain. In the event that the processing architecture does not use an enterprise L-1 data set at the start of its processing chains, the earliest product in the chain will have been populated by the level-0 date of collection, which is where the L-1 MD.Collect.CollctnDate is, in the end, populated from.

MD.Coverage.Temporal.StartDate is derived from the MD.Aggregate.Parent.CollctnDate(j) for its parent, but a level-2 or level-3 product may have been selected as a temporal subset of the data time span of its parent. Thus, to be complete, MD.Coverage.Temporal.StartDate gives the day of collection for the earliest items in the current data set as an

independent item to cover the case that a collection may have begun shortly before midnight on the date of collection start, but the temporal span of the data in the current data set may have a start on the next day. The inheritance of collection date data is illustrated in the case below:

Product in chain	**Level**	**Tag for collection date**
1	1	MD.Collect.CollctnDate ↓
2	2	MD.Aggregate.Parent.CollctnDate(1) ↓
2	2	MD.Coverage.Temporal.StartDate (Where any offset in the starting time that changes the day is accounted for.) ↓
3	3	MD.Aggregate.Parent.CollctnDate(1)
etc	etc	etc

Note also that there is only one start date for data in an enterprise level-2 data set.

MD.Coverage.Temporal.StartDate follows the CMMD pattern of using the Truncated Julian Day (TJD) for expressing a date to year-month-day granularity. MD.Coverage.Temporal.StartDate is the number of days elapsed between midnight on May 24, 1968 and the day (midnight start) of the earliest data populating the current data set.

This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g. http://www.batse.msfc.nasa.gov/tools/date_convert/ or http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via

TJD = INT[JD − 2440000.5]

A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.

Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality

metrics
See Also

20.1.30 MD.Coverage.Temporal.StartTime [L2,3,4,5]

Brief Definition: MD.Coverage.Temporal.StartTime gives the time of the earliest data in the current data set.

indices: none

Units/representation: Seconds, single Float

Treatment/usage: Collection start times for level-2 data sets and above are derived from the parent data set collection start times, but a data set's start time can incorporate an offset from the start of the collected data when a temporal subset of the parent is being selected to create the current data set. MD.Coverage.Temporal.StartTime is the start time of the earliest wideband data for the current data set. This represents the number of SI seconds past midnight on the TJD when the *selection of* collected data for the current data set begins. An implementation of a GUI can display MD.Coverage.Temporal.StartTime as HHMMSS.sssssss, but storing the metadata in seconds units permits easy numerical manipulation.

Single float precision will permit start times to be stated to a precision of at least 0.01 seconds on any day. The lidar mission control system should command the sensor operations to begin at 0.01 start time granularity to assure that the sequencing of collection operations is unambiguously determined by MD. Coverage.Temporal.StartTime.

Together with MD.Coverage.Temporal.StartDate, this determines the day and time of the start of product coverage in the current data set.

Governing quality metrics

See Also: MD.Coverage.Temporal.StartDate MD.Aggregate.Parent.StartTime(j) MD.Collect.StartTime

20.1.31 MD.Coverage.Voids.Atmsphrc(j,k) [L1,2,3]

Brief Definition: MD.Coverage.Voids.Atmsphrc(j,k) gives the vertices for a list of geographic regions within the outer boundary of the data set known to have absent or degraded data due to atmospheric losses.

indices: Index, j, ranges from PD.LoIndx.Voids(1) to PD.HiIndx.Voids(1), that is over the reported voids.
Index, k, ranges from 1 to 8, supporting latitude-longitude pairs for 4 vertices

Units/representation: Degrees, Float

Treatment/usage: MD.Coverage.Voids.Atmsphrc(j,k) describes horizontal regions where

collection is sufficiently degraded due to atmospheric losses (includes the effect of clouds, dust and smoke or other airborne obscurants) that the likelihood of detecting returns from scatterers in the height range required, that is up to MD.Coverage.Bounds.Z(3), is less than 50 percent.

We describe void regions by the circumscribing geodetic quadrilateral of the horizontal footprint of a region of degraded or missing data. Each quadrilateral is given by four vertices supplied in counterclockwise order as viewed from above the earth. The first vertex is the north most or the north-west vertex, which ever is unique.
MD.Coverage.Voids.Atmsphrc(j,1) is the latitude of the North-West {or North most} vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,2) is the longitude of the North-West {or North most}vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,3) is the latitude of the second, e.g. South-West, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,4) is the longitude of the second, e.g. South-West, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,5) is the latitude of the third, e.g. South-East, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,6) is the longitude of the third, e.g. South-East, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,7) is the latitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Atmsphrc(j,8) is the longitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region

Governing quality metrics

See Also PD.HiIndx.Voids(j)

20.1.32 MD.Coverage.Voids.Incompl(j,k) [L1,2,3]

Brief Definition MD.Coverage.Voids.Incompl(j,k) gives the vertices for a list of geographic regions within the outer boundary of the data set known to have only partial data collection due to unavoidable and predictable collection mode behavior.

indices Index, j, ranges from PD.LoIndx.Voids(2) to PD.HiIndx.Voids(2), that is over the reported voids.
Index, k, ranges from 1 to 8, supporting latitude-longitude pairs for 4 vertices and a height

Units/representation Degrees, Float

Treatment/usage MD.Coverage.Voids.Incompl(j,k) population should be limited to the case of

single pass geometric diversity shortfall in collected regions outside the tasked area, inside the region supported by some of the collection interval, but known to be necessarily incomplete in order to achieve full diversity within the tasked area. The beginning and ends of swaths collected with nadir diverse channels will tend not to have been visited by the FOV of all of the channels. These portions must be collected to support full diversity of the inner portions of the swath, but do not meet the full intended geometric diversity of the collection operation. They will not be fully void of points, but do not completely satisfy the collection objective either. They may be left in the product to support downstream processing or supply bonus context for the tasked collection region of the swath.

We describe void regions by the circumscribing geodetic quadrilateral of the horizontal footprint of a region of degraded or missing data. Each quadrilateral is given by four vertices supplied in counterclockwise order as viewed from above the earth. The first vertex is the north most or the north-west vertex, which ever is unique.

MD.Coverage.Voids.Incompl(j,1) is the latitude of the North-West {or North most} vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,2) is the longitude of the North-West {or North most} vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,3) is the latitude of the second, e.g. South-West, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,4) is the longitude of the second, e.g. South-West, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,5) is the latitude of the third, e.g. South-East, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,6) is the longitude of the third, e.g. South-East, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,7) is the latitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region
MD.Coverage.Voids.Incompl(j,8) is the longitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region

Governing quality metrics

See Also PD.HiIndx.Voids(j)

20.1.33 MD.Coverage.Voids.Masked(j,k) [L1,2,3]

Brief Definition MD.Coverage.Voids.Masked(j,k) gives the vertices for a list of geographic regions within the outer boundary of the data set known to be in the lidar

	shadow of terrain features.
indices	Index, j, ranges from PD.LoIndx.Voids(3) to PD.HiIndx.Voids(3), that is over the reported voids due to terrain masking . Index, k, ranges from 1 to 8, supporting latitude-longitude pairs for 4 vertices and a height
Units/representation	Degrees, Float
Treatment/usage	MD.Coverage.Voids.Masked(j,k) gives the regions within a product that are void due to terrain masking. This means that a region of the height range required, that is up to MD.Coverage.Bounds.Z(3), fell within the lidar shadow of a terrain feature due to collection nadir choice. MD.Coverage.Voids.Masked(j,k) population should be limited to single pass coverage within the tasked area where the affected areas are at least 10 percent of the swath width, but known to be necessary consequences of terrain[94] and general collection constraints. Collections with other geometry constraints would be needed to fill such portions of the tasked area. We describe void regions by the circumscribing geodetic quadrilateral of the horizontal footprint of a region of degraded or missing data. Each quadrilateral is given by four vertices supplied in counterclockwise order as viewed from above the earth. The first vertex is the north most or the north-west vertex, which ever is unique. MD.Coverage.Voids.Masked(j,1) is the latitude of the North-West {or North most} vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,2) is the longitude of the North-West {or North most} vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,3) is the latitude of the second, e.g. South-West, vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,4) is the longitude of the second, e.g. South-West, vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,5) is the latitude of the third, e.g. South-East, vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,6) is the longitude of the third, e.g. South-East, vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,7) is the latitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region MD.Coverage.Voids.Masked(j,8) is the longitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region
Governing quality metrics	

[94] That is, low resolution DEMS could be used to predict this in advance of the lidar collection, or post collection analysis can explain the data voids with the terrain masking mechanism.

See Also PD.HiIndx.Voids(j)

20.1.34 MD.Coverage.Voids.UnColl(j,k) [L1,2,3]

Brief Definition MD.Coverage.Voids.UnColl(j,k) gives the vertices for a list of geographic regions within the outer boundary of the data set known to be uncollected bands within the data set.

indices Index, j, ranges from PD.LoIndx.Voids(4) to PD.HiIndx.Voids(4), that is over the reported voids due to intended and predictable lack of sensor contact within the tasked region .
Index, k, ranges from 1 to 8, supporting latitude-longitude pairs for 4 vertices and a height

Units/representation Degrees, Float

Treatment/usage When nadir diversity is collected by a sensor capable of collecting only a single nadir angle at a time, that diversity can be achieved in a single collection by alternating the nadir angle for groups of scans. While the sensor is engaged continuously in time, this mode of collection does not cover the ground contiguously, but rather predictably leaves uncollected gaps. Such a mode would be used in a plan that uses addtional collection operations to collect in the uncollected regions. It is likely anticipated that an enterprise level 4 data set would be created by merging such collects to create a complete geometrically diverse lidar point cloud.
Like MD.Coverage.Voids.Incompl(j,k), MD.Coverage.Voids.UnColl(j,k) documents regions that are void due to the attempt to achieve geometric diversity. The difference between the two is due to the impact of using geometrically diverse channels to scan (.Incompl) vs single channel FOV shifting (.Uncoll).
MD.Coverage.Voids.UnColl(j,k) describes the voids in a data set induced by collecting according to the above conops.

We describe void regions by the circumscribing geodetic quadrilateral of the horizontal footprint of a region of degraded or missing data. Each quadrilateral is given by four vertices supplied in counterclockwise order as viewed from above the earth. The first vertex is the north most or the north-west vertex, which ever is unique.
MD.Coverage.Voids.UnColl(j,1) is the latitude of the North-West {or North most} vertex of the jth void quadrilateral region
MD.Coverage.Voids.UnColl(j,2) is the longitude of the North-West {or North most} vertex of the jth void quadrilateral region
MD.Coverage.Voids.UnColl(j,3) is the latitude of the second, e.g. South-West, vertex of the jth void quadrilateral region
MD.Coverage.Voids.UnColl(j,4) is the longitude of the second, e.g. South-

	West, vertex of the jth void quadrilateral region MD.Coverage.Voids.UnColl(j,5) is the latitude of the third, e.g. South-East, vertex of the jth void quadrilateral region MD.Coverage.Voids.UnColl(j,6) is the longitude of the third, e.g. South-East, vertex of the jth void quadrilateral region MD.Coverage.Voids.UnColl(j,7) is the latitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region MD.Coverage.Voids.UnColl(j,8) is the longitude of the fourth, e.g. North-East, vertex of the jth void quadrilateral region
Governing quality metrics	
See Also	PD.HiIndx.Voids(j) MD.Coverage.Voids.Incompl(j,k)

MD.Dynamics.*

MD.Dynamics.* parameters describe items associated with dataset suitability for temporal dynamic exploitation and is intended for use in data discovery, not exploitation.

20.1.35 MD.Dynamics.Exposure [L2,3,4,5]

Brief Definition	MD.Dynamics.Exposure gives the time to collect one look of a **L**ocal **S**cene patch in the dataset.
indices	None
Units/representation	Seconds, float
Treatment/usage	MD.Dynamics.Exposure is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. MD.Dynamics.Exposure values are not necessarily precise, but should be populated so as to fairly indicate the character of local scene patches in data that may be exploited for its temporal dynamics. To represent the degree to which the dataset renders a scene frame as a frozen instant, the time duration to create a single exposure for one look at the local scene patch is provided. It may be used to determine if the temporal time resolution of the dataset is suitable for the user's needs. Objects moving a significant fraction of the local scene patch size during the time indicated in MD.Dynamics.Exposure would likely yield a smeared signature.
Governing quality metrics	This value should be populated with a value that is within one percent of the average single look exposure time for the dataset.
See Also	

20.1.36 MD.Dynamics.FrameRate [L2,3,4,5]

Brief Definition	MD.Dynamics.FrameRate determines the number of times per second that a **L**ocal **S**cene patch is presented in the dataset.
indices	None

Units/representation	Dimensionless, UInt8
Treatment/usage	MD.Dynamics.FrameRate is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. MD.Dynamics.FrameRate values are not necessarily precise, but should be populated so as to fairly indicate the density of temporal sampling of local scene patches in data that may be exploited for its temporal dynamics. To complement the exposure time given by MD.Dynamics.Exposure , the number of frames per second can be determined from MD.Dynamics.FrameRate. To accommodate systems that may sample a local scene at intervals longer than than a second between exposures, the physical frame rate will be 0.1 x MD.Dynamics.FrameRate local scene patch exposures in any second. Thus, a value of MD.Dynamics.FrameRate equal to 1 would mean an inter-frame interval of 10 seconds, or a frame rate of 0.1Hz. MD.Dynamics.FrameRate=100 means that the frame rate is 10 per second or 10 Hz. A standard video rate of 60Hz would be indicated by MD.Dynamics.FrameRate=600.
Governing quality metrics	The closest (rounded) integer value to ten times the actual frame rate should always be used to populate this term.
See Also	

20.1.37 MD.Dynamics.LS_PanCode(j) [L2,3,4,5]

Brief Definition	MD.Dynamics.LS_PanCode(j) describes the type of panning present in the dataset.
indices	Index, j=1,2
Units/representation	Dimensionless, UInt8
Treatment/usage	MD.Dynamics. LS_PanCode(j) is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. This parameter covers two aspects of how panning was done for the dataset. MD.Dynamics. LS_PanCode(1) =0 indicates that pans scroll through the scene with very small FOV shifts between exposures during a pan action. =1 indicates that pans jump through the scene with FOV shifts comparable to or greater than a local scene patch in size between exposures during a pan action. MD.Dynamics. LS_PanCode(2) is a distortion figure for the scene data during pan actions. The units of MD.Dynamics. LS_PanCode(2) are the number of FOV extents in the panning direction by which data in two parts of the local scene patch are relatively displaced. A value of 0 indicates no distortion and a value of 100 indicates shifts of an entire local scene patch in extent.

Distortion may exceed 100 percent during rapid scanning of certain types of sensors (but this should not be an issue for a sensor whose detections are always represented in the earth frame as is the case for most lidar operation.)

{For lidar, MD.Dynamics. LS_PanCode(2) should always be zero}

Governing quality metrics

See Also

20.1.38 MD.Dynamics.LS_PanRate(j) [L2,3,4,5]

Brief Definition

MD.Dynamics.LS_PanRate(j) gives indicative values of the pan rate for the dataset.

indices

Index j= 1,2,3

Units/representation

Patch extents per second, Float

Treatment/usage

MD.Dynamics. LS_PanRate(j) is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. MD.Dynamics. LS_PanRate values are not necessarily precise, but should be populated so as to fairly indicate the character of local scene patches in data that may be exploited for its temporal dynamics.

If the sensor look direction is panned across the scene being collected, the amount of scene in the FOV of the size of a local scene patch that is shifted between consecutive exposures can be expressed in local scene patch extents per second.To handle the potential for variable panning rates within a single dataset, we define the following:
MD.Dynamics.LS_PanRate(1)= the minimum number of frame extents moved per second during a pan action for the dataset.
MD.Dynamics.LS_PanRate(2)= the maximum number of frame extents moved per second during a pan action for the dataset.
MD.Dynamics.LS_PanRate(3)= the typical number of frame extents moved per second during a pan action for the dataset. By “typical” we mean a plurality of a short list of discrete pan rates or a rate that is not exceeded more than half the time.

MD.Dynamics. LS_PanRate(j) applies both for collections that stare and then pan between dwells and for collections for which the lines of sight is in a continuous scroll over the scene.

Governing quality metrics

See Also

20.1.39 MD.Dynamics.LS_PatchDat(j) [L2,3,4,5]

Brief Definition	MD.Dynamics.LS_PatchDat(j) describes **L**ocal **S**cene patches in the dataset.
indices	Index, j, ranges from 1 to 5
Units/representation	Various, Uint16
Treatment/usage	MD.Dynamics.LS_PatchDat(j) is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. MD.Dynamics.LS_PatchDat(j) values are not necessarily precise, but should be populated so as to fairly indicate the character of local scene patches in data that may be exploited for its temporal dynamics. A "local scene patch" is defined at the spatial extent of scene within a dataset that is repeatedly collected in time intervals that are short compared to the collection duration of the full dataset. A data set's spatial coverage might be large compared to the "local scene patch" or it could be essentially limited to that patch as if the sensor were continuously staring at a fixed scene region. MD.Dynamics.LS_PatchDat(1) is the length in meters of a local scene patch as projected to the ground. It is a horizontal dimension of the scene (volume) and when the aspect ratio of the scene patch horizontal extent is not unity, the length indicated by this parameter corresponds to the longer of the two horizontal dimensions of the local scene patch. For effectively rectangular scene patch shapes, MD.Dynamics.LS_PatchDat(1) is length of the long side. For elliptical patch shapes, MD.Dynamics.LS_PatchDat(1) corresponds to the major axis. MD.Dynamics.LS_PatchDat(2) is the width in meters of a local scene patch as projected to the ground. It is a horizontal dimension of the scene (volume) and when the aspect ratio of the scene patch horizontal extent is not unity, the length indicated by this parameter corresponds to the shorter of the two horizontal dimensions of the local scene patch. For effectively rectangular scene patch shapes, MD.Dynamics.LS_PatchDat(2) is the length of the short side. For elliptical patch shapes, MD.Dynamics.LS_PatchDat(2) corresponds to the minor axis. MD.Dynamics.LS_PatchDat(3) is the extent in meters of a local scene patch along the line of sight. For effectively rectangular scene patch shapes, and vertical viewing, MD.Dynamics.LS_PatchDat(3) is the height of a represented cubical scene volume. For 3-D imaging, MD.Dynamics.LS_PatchDat(3) is finite and is generally a number much less than 65535 meters . For data that represents a 2-D view of the scene, effectively the full line of sight is integrated into each horizontal scene sample

and this is indicated by giving MD.Dynamics.LS_PatchDat(3) a value of 0.

MD.Dynamics.LS_PatchDat(4) indicates the best potential linear sampling density in the direction orthogonal to the line of sight for the dataset. The phrase "best potential" indicates that the sensor system and subsequent processing would retain this much detail, if it were present in the scene. The absence of scatterers in any region of the scene is not accounted for by this parameter. MD.Dynamics.LS_PatchDat(4) determines the number of samples per meter. If the sampling density were not isotropic, it would be determined by the geometric mean of sample spacing in the directions of minimum and maximum sample spacing. In order to accommodate sampling densities less than 1 per meter and because sampling at densities exceeding 500 per meter is not practical for geospatial products, MD.Dynamics.LS_PatchDat(4) is given as an integer whose value is 100 times the sampling density in meters {sample spacing in meters =100/ MD.Dynamics.LS_PatchDat(4) }. Thus, MD.Dynamics.LS_PatchDat(4) would be 1 if the sample spacing were 100 meters, 10 if the sample spacing were 10 meters, and 1000 if the sample spacing were 10 centimeters. Larger values indicate "better" scene fidelity.

MD.Dynamics.LS_PatchDat(5) encodes how a local patch is scanned by the sensor.

> MD.Dynamics.LS_PatchDat(5)=0 indicates that collection of a local patch does not entail scanning at all. Effectively, this indicates use of a staring array in the sensor focal plane or its equivalent.
> MD.Dynamics.LS_PatchDat(5)=1 indicates raster scanning in a plane orthogonal to the line of sight.
> MD.Dynamics.LS_PatchDat(5)=2 indicates Lissajou scanning to cover a local scene patch.

MD.Dynamics.LS_PatchDat(6) provides the size of a local scene patch as a percentage of the total dataset FOV. If the dataset were collected in a single long stare, MD.Dynamics.LS_PatchDat(6) would be 100. If a local scene patch is one hundredth of the area presented in the dataset, MD.Dynamics.LS_PatchDat(6) would be 1. The maximum permitted value of MD.Dynamics.LS_PatchDat(6) is thus 100 and if it is given a value of 0, that indicates the full FOV exceeds 100 times the area of a local scene patch.

Note, this information is for data selection purposes and is not intended to be precise values as would support detailed exploitation. Most dynamically oriented data might be initially exploited in real time, but these data items are

of value for both real time and archival exploitation.

Governing quality metrics: Fair representation may be take to be values whose errors are within 25 to 50% of precise values. A factor of 2 or more error would be of concern.

See Also:

20.1.40 MD.Dynamics.LS_PatchDwell(j) [L2,3,4,5]

Brief Definition: MD.Dynamics.LS_PatchDwell characterizes how many seconds **L**ocal **S**cene patches persist in the dataset.

indices: Index, j=1,2,3

Units/representation: Seconds, Float

Treatment/usage: MD.Dynamics. LS_PatchDwell is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. MD.Dynamics. LS_PatchDwell values are not necessarily precise, but should be populated so as to fairly indicate the character of local scene patches in data that may be exploited for its temporal dynamics. If the dataset is for a case where the the sensor is in a fairly continuous scroll over the scene, a dwell is defined as the duration of time for which 50 percent of the scene in an initial local scene patch remains in the product's single look field of view. If we stare and pan between looks, MD.Dynamics.LS_PatchDwell is the time between panning intervals. In particular, the duration of dwell for different local scene patches or visits to a single local scene patch may vary. To indicate the variation of local scene dwell we define the following:

MD.Dynamics. LS_PatchDwell(1)= the minimum number of seconds for a dwell on a local scene patch for the current dataset.

MD.Dynamics. LS_PatchDwell(2)= the maximum number of seconds for a dwell on a local scene patch for the current dataset.

MD.Dynamics. LS_PatchDwell(3)= the mean number of seconds for a dwell on a local scene patch for the current dataset.

Governing quality metrics:

See Also:

20.1.41 MD.Dynamics.VisitsPerPatch(j) [L2,3,4,5]

Brief Definition: MD.Dynamics.VisitsPerPatch(j) indicates the number of local region revisits that may occur within the dataset.

indices: Index, j=1,2,3

Units/representation: Dimensionless, UInt8

Treatment/usage: MD.Dynamics.VisitsPerPatch(j) is one of a suite of data discovery level parameters that indicate the degree of temporal scene dynamics rendered in the dataset. MD.Dynamics VisitsPerPatch(j) values are not necessarily

precise, but should be populated so as to fairly indicate the character of local scene patches in data that may be exploited for its temporal dynamics.

If the sensor stares only, the number of visits for a local scene patch will be just one for a dataset. However, if the line of sight pans away from a local scene patch and returns for another dwell later, each dwell is called a visit. The number of times the scene region for a local scene patch is visited may vary due to active panning of the scene during collection. To characterize the number of visits to patches in a dataset, we define the following:
MD.Dynamics.VisitsPerPatch(1)= minimum number of visits to any local scene patch and must have a value of at least one in any dataset.
MD.Dynamics.VisitsPerPatch(2)= maximum number of visits to any local scene patch and must have a value of at least one in any dataset.
MD.Dynamics.VisitsPerPatch(3)= the typical number of visits to any local scene patch and must have a value of at least one in any dataset. By "typical", we mean a plurality value for the number of visits to any local scene patch whenever the value exceeds 30% of the maximum. Otherwise we use a 50th percentile value.

Governing quality metrics
See Also

MD.File.*

MD.File.* parameters describe items associated with security handling of the data set file. This includes the range of classification markings found on document classification banners and in classification authority blocks. A subset of these markings is applicable to any one data set. The applicable Security Classification Guide must be consulted for appropriate data set classification marking.

Security markings are governed by normative references that themselves will be subject to revision and re-publication. This CMMD will be included in a process that the GWG is adopting for maintaining harmonization into the future among documents governing resources and datasets that will have security marking information included. The plan is for this information to be eventually removed from this CMMD and for implementers to use what will be registered names when populating security marking fields in name-value supporting data containers in the manner described in this CMMD. For completeness during the interval before this process is fully in force, security marking information is defined below.

20.1.42 MD.File.Class[L1,2,3,4,5]

Brief Definition	MD.File.Class contains the security classification level of metadata and collected sensor product data when presented together.
indices	None
Units/representation	Security level code, BCS character string, Length 8.
Treatment/usage	This field shall contain a value representing the security classification level of the entire file, wideband and narrowband components included. Typically a library function may strip off and save the narrowband "metadata" for use in building directories and other discovery services. That process may lead to different classification of the separated parts of the complete file. This item applies only to the data set in its entirety. The MD.File.Class character string should be post padded with blanks as appropriate. Valid values are: TS = Top Secret, S = Secret, C = Confidential, R = Restricted, U = Unclassified. CTS = Cosmic Top Secret CTS-B = Cosmic Top Secret - Bohemia CTS-BALK = Cosmic Top Secret - Balk NU = NATO Unclassified NR = NATO Restricted NC = NATO Confidential NS = NATO Secret CTSA = Cosmic Top Secret Atomal NSAT = NATO Secret Atomal NCA = NATO Confidential Atomal
Governing quality metrics	The classification represented by the value of MD.File.Class must be correct.
See Also	MD.Meta.Class

20.1.43 MD.File.ClassBy[L1,2,3,4,5]

Brief Definition	MD.File.ClassBy identifies the original classification authority for the content of the data set.
indices	None
Units/representation	Dimensionless, BCS Character String, length 8
Treatment/usage	Reference the applicable Security Classification Guide. This field is used to support labels for displayed versions of the data, labeling of derived products, and to support management of classification changes for the data in the data

set. Typical values indicate that the director of the Government agency responsible for the creation of the data product is the original classification authority. Thus a value of "D/NGA " might be assigned to many classified products. The MD.File.ClassBy character string should be post padded with blanks as appropriate.

Governing quality metrics

See Also

20.1.44 MD.File.Comp[L1,2,3,4,5]

Brief Definition: MD.File.Comp identifies special handling caveats.

indices: None

Units/representation: Character String, BCS length 14

Treatment/usage: Reference the applicable Security Classification Guide. This field permits spelling out the control system for certain data sets. Those responsible for populating and adhering to the constraints of this field will be instructed on its detailed interpretation outside of the control of this document.

Governing quality metrics

See Also

20.1.45 MD.File.DeclDate[L1,2,3,4,5]

Brief Definition: MD.File.DeclDate identifies the declassification date.

indices: None

Units/representation: Days, UInt16

Treatment/usage: Reference the applicable Security Classification Guide.
is the number of days elapsed between midnight on May 24, 1968 and the declassification date for the current data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest.

It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality metrics

See Also

20.1.46 MD.File.DeclExemtn[L1,2,3,4,5]

Brief Definition: MD.File.DeclExemtn identifies applicable declassification exemption.

indices: None

Units/representation: BCS Character String Length (8),

Treatment/usage: Reference the applicable Security Classification Guide. There is no currently known reason this would be populated for lidar data products, but is included for consistency of the metadata group and potential future use. Some documents are exempt from declassification and must be so marked when applicable.

Governing quality metrics

See Also

20.1.47 MD.File.DeclMeth[L1,2,3,4,5]

Brief Definition: MD.File.DeclMeth contains declassification method instructions.

indices: None

Units/representation: BCS Character String, Length 10

Treatment/usage: Reference the applicable Security Classification Guide. This string carries a designation for the instruction method for declassifying. This item would only be populated when it is known at product generation time that only one method of declassification is acceptable. Otherwise, one of several methods known to the control system for the product could apply and need not be documented within the lidar product.

Governing quality metrics

See Also

20.1.48 MD.File.DissContrls(j) [L1,2,3,4,5]

Brief Definition: MD.File.DissContrls(j) contains dissemination control codes.

indices: Index, j, =1,2

Units/representation: Character String (20), BCS pad with trailing blanks

Treatment/usage: Reference the applicable Security Classification Guide/addendum for code values.

MD.File.DissContrls(1) carries an alpha string indicating distribution or release limitations.

MD.File.DissContrls(2) carries a numeric code applicable for certain values of MD.File.DissContrls(1). When no numeric code applies, but there is an alphabetic string populating MD.File.DissContrls(1), MD.File.DissContrls(2) should be filled with blank characters.

Governing quality metrics

See Also

20.1.49 MD.File.DG[L1,2,3,4,5]

Brief Definition: MD.File.DG contains downgrade instructions

indices: None

Units/representation: BCS Character String, Length 20

Treatment/usage: Reference the applicable Security Classification Guide. This would be populated in accordance with the downgrading policies of the generating system/organization. This is intended to cover authorized processes such as "Downgrade to C when …", "DG to S by manual review" but not provide target date information for downgrading. If MD.File.DG is not populated, but MD.File.DG_Date is, then automatic downgrading is understood to be scheduled. A change in downgrade policy for the product subsequent to its generation would require editing the file to update this field.

Governing quality metrics

See Also: MD.File.DG_Date

20.1.50 MD.File.DG_Date[L1,2,3,4,5]

Brief Definition: MD.File.DG_Date contains a downgrade Date

indices: None

Units/representation: Days, UInt16

Treatment/usage: Reference the applicable Security Classification Guide.
is the number of days elapsed between midnight on May 24, 1968 and the scheduled date of downgrading for the current data set. If MD.File.DG is not populated, but MD.File.DG_Date is, then automatic downgrading is understood to be scheduled. A change in downgrade policy for the product subsequent to its generation would require editing the file to update this field.

This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g. http://www.batse.msfc.nasa.gov/tools/date_convert/ or http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via

TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality metrics

See Also

20.1.51 MD.File.Reason[L1,2,3,4,5]

Brief Definition: MD.File.Reason contains the reason(s) for classification.

indices: None

Units/representation: BCS Character String, Length 15

Treatment/usage: Reference the applicable Security Classification Guide. This string carries the applicable list of reason codes for classification of the content. Post pad with blanks as needed.

Governing quality metrics

See Also

MD.InfAssur.*

MD.InfAssur.* parameters describe items associated with information assurance of the integrity of the file in which they are embedded.

20.1.52 Candidate MD.InfAssur.Seal [L1,2,3,4,5]

Brief Definition: MD.InfAssur.Seal is an information assurance parameter meant to act as a security seal that can be used to detect alterations to a CMMD compliant file after authoritative generation.

indices: none

Units/representation: None/ 256 bit string

Treatment/usage: MD.InfAssur.Seal acts as an imbedded security seal. The data description provides the space for this data but does not constrain the implementation. Typically this item would be located near the front of the CMMD compliant file and might contain a hash of the trailing portion of the file which would be registered with an information assurance site accessible by intended users of the CMMD compliant product.

Governing quality metrics: Strict compliance to Doc

See Also

20.1.53 Candidate MD.InfAssur.NonRepud [L1,2,3,4,5]

Brief Definition	MD.InfAssur.NonRepud is an information assurance parameter meant to act as a non-repudiation guarantee that can be used to authenticate the originator's assurance of file origin.
indices	none
Units/representation	None/ 256 bit string
Treatment/usage	MD.InfAssur.NonRepud acts as an imbedded security seal. The data description provides the space for this data but does not constrain the implementation. Typically this item would be located near the front of the CMMD compliant file and might contain a (hash of the) digital signature which would be registered with an information assurance site accessible by intended users of the CMMD compliant product.
Governing quality metrics	Strict compliance to Doc
See Also	

MD.LegalConstr.*

MD.LegalConstr.* parameters describe the rights, license restrictions and secondary use restrictions associated with non-governmental ownership of data in a data set.

20.1.54 MD.LegalConstr.AttributionCitation[L1,2,3,4,5]

Brief Definition	MD.LegalConstr.AttributionCitation indicates who/what must be credited in derivative use of the content of the current product governed per the originator's intellectual or physical property rights.
indices	None
Units/representation	BCS Character String, Length 60
Treatment/usage	MD.LegalConstr.AttributionCitation is given a value that acts as a citation attribution that identifies the holder of a patent, copyright, or proprietary rights. This string contains text that would be pasted into derivative products to act as appropriate attribution so that plagarism is not committed against holders of legal rights to data in the data set.
Governing quality metrics	
See Also	

20.1.55 MD.LegalConstr.Derivation(j) [L1,2,3,4,5]

Brief Definition	MD.LegalConstr.Derivation(j) is a vector of codes for applicable legal protection governing access, use, secondary distribution

	(copyright, license, trademark, patent,…)
indices	j = 1 to 5, non-blank values populated into the number of coded derivation sources that actually apply.
Units/representation	BCS character string, Length 15
Treatment/usage	MD.LegalConstr.Derivation(j) carries a list of character strings that are labels or codes for reasons that there are legal constraints on the use of content in the file (resource). Multiple values may be in effect for any given data product, thus this is a data vector. If fewer than 5 reasons for constraint in derivative use of the data in other products are in effect, the remaining elements of MD.LegalConstr.Derivation(j) should be blank filled. Potential constraint derivations include: Copyright, patent, patentPending, trademark, license, restricted
Governing quality metrics	
See Also	MD.LegalConstr.Dissem(j)

20.1.56 MD.LegalConstr.Dissem(j) [L1,2,3,4,5]

Brief Definition	MD.LegalConstr.Dissem(j) is a vector of codes for distribution constraints applicable to the current data set or its derived products.
indices	j = 1 to 5, non-blank values are populated into the number of coded distribution constraints that actually apply.
Units/representation	BCS character string, length 15
Treatment/usage	MD.LegalConstr.Dissem(j) carries a list of character strings that are labels or codes for the legal constraints on use of content in the file (resource). Mulitple values may be in effect for any given data product, thus this is a data vector. If fewer than 5 constraints in derivative use of the data in other products are in effect, the remaining elements of MD.LegalConstr.Derivation(j) should be blank filled. Potential constraints labels include: Attrib'n req'd, permiss'n req'd, Fee required, gov't use only, non-profit use, RB *YYYYMMDD* (RB➔ restricted before), RA *YYYYMMDD* (RA➔ restricted after).
Governing quality metrics	
See Also	MD.LegalConstr.Derivation(j)

MD.Meta.*

MD.Meta.* parameters describe the items associated with documentation of metadata for this data set.

20.1.57 MD.Meta.Class[L1,2,3,4,5]

Brief Definition	MD.Meta.Class identifies the highest classification of any meta data item

	when the wideband content is not present.
indices	None
Units/representation	None
Treatment/usage	Security level code, BCS character string, Length 8.
Governing quality metrics	This field shall contain a value representing the security classification level of only the metadata (narrow band data) in the file. Typically, a library function may strip off and save the narrowband "metadata" for use in building directories and other discovery services. That process may lead to different classification of the separated parts of the complete file. This item applies only to the portion that would be used to document the file pedigree and support its discovery. This is information that is likely to be retained by the discovery infrastructure of a library function. The MD.Meta.Class character string should be post padded with blanks as appropriate. Valid values are: T = Top Secret, S = Secret, C = Confidential, R = Restricted, U = Unclassified. CTS = Cosmic Top Secret CTS-B = Cosmic Top Secret - Bohemia CTS-BALK = Cosmic Top Secret - Balk NU = NATO Unclassified NR = NATO Restricted NC = NATO Confidential NS = NATO Secret CTSA = Cosmic Top Secret Atomal NSAT = NATO Secret Atomal NCA = NATO Confidential Atomal
See Also	MD.File.Class

20.1.58 MD.Meta.DAL_VersionName(j) [L1,2,3,4,5]

Brief Definition	MD.Meta.DAL_VersionName(j) is a list of strings indicating the set of data access library (DAL) modules that can read this file.
indices	Index, j, ranges from PD.LoIndx.DAL to PD.HiIndx.DAL, that is over the number of DAL routines known to successfully read current file provided in the current data set.
Units/representation	BCS string, Length 10
Treatment/usage	The DAL module that was used to create the current data set is assured of being able to also read the current data set. For that reason, its name is

incorporated into the file name when the data set is originally generated and is available for use by downstream software before the file is ever read. However, it may be the case that other DAL modules (i.e., instantiations or versions) could be known to also successfully read the file. It is too burdensome to put all of those DAL module names into the file name, but as the library could be responding to requests by users/'using software' that reveal the DAL modules that are available as readers, it is beneficial architecturally that the Libraries know whether their subsequent dissemination of a data set will be compatible with the user's tools as is. The Library would then have a policy of either 1) giving out N/A notices when the users DAL can't read the file as stored, or 2) the Library could perform a conversion to a DAL that the user has access to, or 3) the Library could down load to the requestor of a data set a DAL module that would work with the file that the Library is sending to the user. MD.Meta.DAL_VersionName(j) provides a convenient way for the facility that generates a data set to let the libraries that receive that data set know the reading options that may exist. The strings that populate the data vector, MD.Meta.DAL_VersionName(j), are of the same same form as the DAL synch substring that appears in the data set name, MD.Meta.DS_Name, with the exception that the delimiting "." is omitted. An entry in DAL_VersionName(j) always begins "DAL". The remaining 7 characters [should be registered strings] indicating a unique set of DAL linkable code modules that will successfully ingest data written to the data set. A valid example (once registered) of an entry in MD.Meta.DAL_VersionName(j), is "DALHDF5v02"

Governing quality metrics

See Also MD.Meta.DS_Name

20.1.59 MD.Meta.DR_ContactAddr[L1,2,3,4,5]

Brief Definition MD.Meta.DR_ContactAddr contains a URL where open discrepancy reports (DRs) for the current data set can be viewed or new DRs can be submitted

indices None

Units/representation BCS String, Length 50

Treatment/usage This string, when typed into a web browser, will act as the URL to a website that will accept discrepancy reports related to the current data set. For URLs shorter than 50 characters, this item will be post padded with blanks.

Governing quality metrics

See Also

20.1.60 MD.Meta.DS_Name [L1,2,3,4,5]

Brief Definition (U) MD.Meta.DS_Name is the formal name for the current data set.
indices None
Units/representation Dimensionless, BCS Character String, Length 72
Treatment/usage (U) This document addresses product type level **LID**ar **L**evel **2**, which is also part of the data set name.

An enterprise level-2 data set name is given by concatenating the following substrings:

date/time(TJD based)-- YYYYMMDDHHMMSS.S (16 characters) of file creation
platform identifier--(6 characters)
Collection Num -- (4 characters)
Sensor Identifier --(6 characters)
Channel identifier --(2 characters)
Prod type -- "LIDL2"
Level-n stage/Form –(2 characters)
Generation Site ID –(6 characters)
Spacer string "___" (4 characters)
GenSite Unique product serial number—(up to 10 characters)
DAL synch string ".*CCCCCCCCCC*" e.g ".DALHDF5v02" (11 characters)

In the above described substrings:
Platform identifier--(6 characters) refers to a registered platform identifier that uniquely determines the kind and identity of the lidar bus vehicle. The identifier might be composed of 3 char alphanumeric symbol such as "DC3" or "707" and a tail identifier composed of 3 alpha-numerics such as "012" or "QRP" or "N32". This also permits designating using US N-numbers up N999ZZ as platform identifiers.

Collection Num-- (4 characters) This is a place to composite strings such as a sortie number and a project number. For example, two alpha numeric characters each can be used for a sortie and project to build the collection number. May only be unique for a single day, week, month,…

Sensor Identifier --(6 characters) refers to a registered sensor identifier such as "OPK015", this is a unique tag for the individual lidar instrument.

Channel identifier --(2 characters) This is most readily treated as a number such as "01" or "02". If a sensing instrument has more than a single sensing

channel and can generate multiple wide band data streams, they should be identified by a sequential number starting with 1 and counting upward. If multiple channels are present in a data set, the Channel identifier should be set to “00”. If there is only one channel in the sensor, the value should be set to “01”. The number of channels in a lidar is not large and the 2 digit limit is quite generous.

Prod type The caps fields listed in this data definition, e.g., "LIDL2" should be used as the value of the substring to be concatenated.

Level-n stage/Form –(2 characters) A registered code for the product stage should be used here. Examples of what will be proposed for codes are:

Level-2 forms

S1- noisy point cloud
2a- Noise filtered point cloud using a’priori methods such as coincidence processing of some kind, no scan to scan registration
2b- Scan to scan registered, no noise removal
(no 2c- Scan to scan registered with noise removal w/o waveform clips. That would be level-3)
S2-Waveform clips only
S3- noisy Point clouds with waveform clips
S4- de-noised points with waveform clips
4a- denoised, scan to scan registered points with waveform clips

This document does not govern any derived products. Thus, such products as ‘single collect, TIN with features’, ‘single collect, TIN Bare earth’, ‘single collect , Bare earth DEM’, and ‘Dictionary based Feature map’ are not given processing stage labels here. However, if the standardization stakeholders for derived products were to adopt the method for DS_Name generation advanced here, there could be products generated with names that appeared much like the …DS_Name string defined here but having codes for TINs and DEMs in them

Generation Site ID- This is a registered 6 character identifier for authorized processing sites within the NSG that are authorized lidar processing sites. Examples of strings could be, “NAT002”, “GIAT03”, “EUT005”, The identifiers are alpha-numeric and unique for each processing site

GenSite Unique product serial number—Each generation site creates a serial sequence that is unique within a period of time that can be encoded as an interval beginning on a value of YYYYDDD. Up to 10 characters can be

used and the active part of any string using less than all 10 should have the needed number of zeros pre-pended to created the 10 character serial number.

DAL synch string--- The 11 character DAL synch string is recognizable since it always begins ".DAL" The remaining 7 characters [should be registered strings] indicating a unique set of DAL linkable code modules that will successfully ingest data written to the data set. The (in this case, lidar) data set writing application chooses the registered DAL implementation to use when it outputs the data set. It then places the registered string for the intended DAL ingest module set at the end of the parameter whose name is MD.Meta.DS_Name and gives the output file the same name as the value of MD.Meta.DS_Name.

Example MD.Meta.DS_Name value:
20110811221012.1B47003**1102**LKA01201LIDL2S3GIAT03____0000012345.DALHDF5v02

This data set name is a label designed for uniqueness and should not be parsed for file content information. Direct use of CMMD defined metadata fields within the data set is the correct way to obtain that information. Limited human recognizable content is incorporated for facilitation of exploitation work flow, but this data item is not a replacement for actual *Discovery* or *Exploitation* support data.

This metadata item is analogous to NITF's FTITLE, in that it is a name for the data set that is embedded in the data set. Operating systems may append their own suffixes to the data set name, but this acts as a global core string convention for naming LIDL*n* sets.

Governing quality metrics: Strict compliance to Doc

See Also: MD.Platflorm.ID

20.1.61 MD.Meta.ID[L1,2,3,4,5]

Brief Definition: MD.Meta.ID indicates the specification or standard governing the current data set definitions and formats.

indices: None

Units/representation: BCS string, Length 120

Treatment/usage: MD.Meta.ID carries the name of the approved governing document which describes the data and its structure in the current data set. An example might be: "lidar Data and Metadata: Conceptual Model and Data Dictionary for Volume Return Products: MIL-STD-9999A"
Post pad the name of the governing data standard or product spec document with blanks as necessary. If the name exceeds 120 characters, it should be

converted to a trimmed string limited to those portions of the full title essential to recognizing it uniquely in a list of similar governing documents.

Governing quality metrics

See Also

20.1.62 MD.Meta.Language[L1,2,3,4,5]

Brief Definition: MD.Meta.Language carries the ISO 3 letter code for the language of any text content within the metadata of the current data set.

indices: None

Units/representation: BCS string, Length 3

Treatment/usage: MD.Meta.Language carries a value from the ISO 639-2 language code list represented in Latin alphabet characters. Character string data in the current data set that are in a natural language and do not carry defined codes will be in the language indicated by the MD.Meta.Language code.

Governing quality metrics

See Also

20.1.63 MD.Meta.MaintOrg[L1,2,3,4,5]

Brief Definition: MD.Meta.MaintOrg is the contact information for the organization responsible for maintaining the metadata model and definitions for the current data set.

indices: none

Units/representation: Dimensionless, BCS character string, Length 80

Treatment/usage: MD.Meta.MaintOrg carries a natural language character string with the URL for a site through which a user of a CMMD compliant data set can identify and contact the organization responsible for maintaining the metadata model and definitions used in the data set. This link must be redirected as necessary and kept up to date so that the value populated into any given data set is not perishable and the information will be current as of the time of the access to the site.

Governing quality metrics

See Also: MD.Meta.DR_ContactAddr

MD.NomGeo.*

MD.NomGeo.* parameters describe the items associated with characterizing the degree to which the data set contains data collected with a variation in viewing angle.

20.1.64 MD.NomGeo.Pt.Az.Ctr [L1,2,3,4,5]

Brief Definition: MD.NomGeo.Pt.Az.Ctr is the central value of the collected range of azimuth

angles on a point basis and is intended to characterize point level collection geometry aspect.

indices: none

Units/representation: Degrees, float

Treatment/usage: Diversity of the collection geometry in remote sensing can result from both

- the intended variation of the look direction towards scene positions to assure a variety of collected aspects for individual points, or
- the natural consequence of collecting a wide area of view, especially from a moving sensor.

We use MD.NomGeo.Pt.* to collect nominal values characterizing diversity of collection geometry at the point level so that we can distinguish data collected for the first case above from that collected for the second case, and also to reveal the degree of diversity at the point level to compare with the diversity over the product as a whole. These values are nominal in the sense that they are derived from mission planning data, coarse sensor look direction and platform data or other simple means for the purpose of characterizing the diversity of collection geometry at individual positions in the collected scene. These are not statistics that require quantitative assessment of each point in a data set. When the relative collection orientation of the sensor and target points within the product has variation over the full swath product, the values that populate MD.NomGeo.Pt.* should represent averages over the collected product swath.

Consider a single point in the vicinity of the surface of the earth where a scene to be probed by the sensor lies. Now, viewing the sensor positions from the target point over the full time period that the sensor is collecting data at the location of the point (i.e over the "dwell"), a volume "wedge" is traced out by the plane of incidence. For off nadir collection, there is a range of azimuth to this wedge. As the azimuth increases from a fraction of a degree to full circle (360 degrees), the center of the wedge in azimuth goes from clearly defined as the middle of some azimuth range (such as 10 degrees being the middle of a collection path that moved from 5 degrees to 15 degrees from North) to ambiguous. When a full circle or more is traversed by the sensor as seen from the target point position, we no longer necessarily can say that half the sum of the minimum and maximum azimuths gives the center azimuth angle. In particular, 360 degrees is the same as zero degrees, so half of 0 and half of 360 become numerically ambiguous (and inconsistent) prescriptions. We arbitrarily define the center in azimuth of a collection that is greater than or equal to a full circle of collection to be 0 degrees. No consideration of a partial overage in the number of times the sensor orbits the target point is considered in assigning a center to the collection path in the azimuth direction. Note that if collection is strictly limited to perfect nadir

alignment with respect to the scene vertical axis, the azimuth angle of collection is undefined. In this rare case, the value of azimuth is reported to be -3.60E02 degrees.

MD.NomGeo.Pt.Az.Ctr is the center of the azimuth range from which a position in the scene has been collected, subject to the above clarifications and conventions. When the center of azimuth collection varies over a full product's swath, MD.NomGeo.Pt.Az.Ctr should be the modeled average center azimuth over the full data contained in the data set.

lidar specific note: When precise data about collection geometry for a particular point in a data set is desired, SC.Pulse.SensorPos.*TileNum*(ip,j) data should be used instead of MD.NomGeo.Pt.*.

Governing quality metrics

See Also SC.Pulse.SensorPos.*TileNum*(ip,j) MD.NomGeo.Swath.Az.Ctr

20.1.65 MD.NomGeo.Pt.Az.Spread [L1,2,3,4,5]

Brief Definition MD.NomGeo.Pt.Az.Spread is the range of azimuth angle about the central value of the collected azimuth angles for each point.

indices none

Units/representation Degrees, Float

Treatment/usage MD.NomGeo.Pt.Az.Spread characterizes the point level diversity of azimuth collection in a data set. This is a nominal value obtained by modeling the average over the swath covered in the data set of the individual point azimuth collection interval widths. Note that if collection is strictly limited to perfect nadir alignment with respect to the scene vertical axis, the azimuth angle of collection is undefined. In this rare case, the value of azimuth is reported to be -3.60E02 degrees, and the spread in MD.NomGeo.Pt.Az.Spread is reported to be 0.

MD.NomGeo.Pt.Az.Spread distinguishes a scan and flight line strategy meant to gather azimuth diversity at each point in a single collection from a scan strategy and flight line choice which incidently creates some diversity across points in order to cover area. When the point level azimuth collection aspect sweep varies over the points in a data set, the swath mean of the azimuth collection interval width as modeled from scan and flight line parameters or the tasking data should be used. So a narrow range of azimuth angles may apply to each scene point within a single scan. But re-scanning the point in the same collection operation may bring diversity of azimuth angle. So, if the first scan's average for a point has nominal azimuth angle of 1° (look within a narrow wedge) and a second scan average for same point is 45 degrees the

Pt. mean is 46/2 =23°. The minimum azimuth is 1 and the maximum is 45. This is a diversity collection stategy for which MD.NomGeo.Pt.Az.Spread would be set to 45-1=44 degrees.
If there is only a first scan, a scene position at mid-scan might see the range 0 to 3°. The point center of azimuth is, perhaps, 1.5^{o} and MD.NomGeo.Pt.Az.Spread would be set to 3°.
When a full circle or more has been collected in azimuth, the reported value of MD.NomGeo.Pt.Az.Spread should be 3.60E+02.

Governing quality metrics

See Also MD.NomGeo.Pt.Az.Ctr

20.1.66 MD.NomGeo.Pt.Nadir.Ctr [L1,2,3,4,5]

Brief Definition MD.NomGeo.Pt.Nadir.Ctr is the central value of the collected range of nadir angles on a point basis and is intended to characterize point level collection geometry aspect.

indices none

Units/representation Degrees, float

Treatment/usage Diversity of the collection geometry in remote sensing can result from both

- the intended variation of the look direction towards scene positions to assure a variety of collected aspects for individual points, or
- the natural consequence of collecting a wide area of view, especially from a moving sensor.

We use MD.NomGeo.Pt.* to collect nominal values characterizing diversity of collection geometry at the point level so that we can distinguish data collected for the first case above from that collected for the second case, and also to reveal the degree of diversity at the point level to compare with the diversity over the product as a whole. These values are nominal in the sense that they are derived from mission planning data, coarse sensor look direction and platform data or other simple means for the purpose of characterizing the diversity of collection geometry at individual positions in the collected scene. These are not statistics that require quantitative assessment of each point in a data set. When the relative collection orientation of the sensor and target points within the product has variation over the full swath product, the values that populate MD.NomGeo.Pt.* should represent modeled averages over the collected product swath.
Consider a single point in the vicinity of the surface of the earth where a scene to be probed by the sensor lies. Now, viewing the sensor positions from the target point over the full time period that the sensor is collecting data at the location of the point, a volume “wedge” is traced out by-that part of the

plane of incidence between the ground and the instantaneous line of sight. There is a range of nadir angle to this wedge. As the nadir range increases from a fraction of a degree (nearly straight down viewing) to 180 degrees(looking straight up) the center of the wedge in Nadir remains clearly defined as the middle of the nadir range (such as 10 degrees being the middle of a collection path that moved from 5 degrees to 15 degrees from vertical.) as opposed to the case of azimuth, where there is the 360 degree wrap issue. {We can alternately consider a (possibly irregular) cone about the vertical at at an observed point that is traced out by the line of sight. There will be some range from the vertical axis (nadir angle=0) to the greatest collected nadir angle }

MD.NomGeo.Pt.Nadir.Ctr is the center of the nadir angle range from which a position in the scene has been collected. When the center of collection nadir varies over a full product's swath, MD.NomGeo.Pt.Nadir.Ctr should be the modeled average center position over the full swath contained in the data set. Typically, if the sensor is well elevated above a scene, varying the look direction can introduce point level nadir diversity. If the sensor is at a fixed ground level position, point level nadir diversity is not achievable, but swath level nadir diversity could be substantial depending on the field of view collected.

lidar specific note: When precise data about collection geometry for a particular point in a data set is desired, SC.Pulse.SensorPos.*TileNum*(ip,j) data should be used instead of MD.NomGeo.Pt.*.

Governing quality metrics: Values must be populated such that the cosine of the nadir angle is within 1% of the ideal/error-free value.

See Also: MD.NomGeo.Swath.Nadir.Ctr

20.1.67 MD.NomGeo.Pt.Nadir.Spread [L1,2,3,4,5]

Brief Definition: MD.NomGeo.Pt.Nadir.Spread is the range of nadir angle about the central value of the collected nadir angles for each point.

indices: none

Units/representation: Degrees, Float

Treatment/usage: MD.NomGeo.Pt.Nadir.Spread characterizes the point level diversity of nadir angle collection in a data set. This is a nominal value obtained by modeling the average over the swath covered in the data set of the individual point nadir collection interval widths.

MD.NomGeo.Pt.Nadir.Spread distinguishes a scan and flight line strategy meant to gather nadir diversity at each point in a single collection from a scan strategy and flight line choice which incidently creates some diversity across points in order to cover area. When the point level nadir collection aspect sweep varies over the points in a data set, the swath mean of the nadir

collection interval width as modeled from scan and flight line parameters or the tasking data should be used. So a narrow range of nadir angles may apply to each scene point within a single scan. But re-scanning the point in the same collection operation may bring diversity of nadir angle. So, if the first scan's average for a point has nominal nadir angle of 1° (look within a narrow wedge) and a second scan average for the same point is 45 degrees, the Pt. mean is 46/2 =23°. The minimum nadir is 1 and the maximum is 45 degrees. This is a diversity collection stategy for which MD.NomGeo.Pt.Nadir.Spread would be set to 45-1=44 degrees.
If there is only a first scan, a scene position at mid-scan might see the range 0 to 3°. The point center of nadir is, perhaps, 1.5^o and MD.NomGeo.Pt.Nadir.Spread would be set to 3°.

Governing quality metrics

See Also MD.NomGeo.Swath.Nadir.Spread MD.NomGeo.Pt.Nadir.Ctr

20.1.68 MD.NomGeo.Swath.Az.Ctr [L1,2,3,4,5]

Brief Definition MD.NomGeo.Swath.Az.Ctr is the central value of the collected range of azimuth angles for the swath

indices none

Units/representation Degrees, Float

Treatment/usage Diversity of the collection geometry in remote sensing can result from both

- the intended variation of the look direction towards scene positions to assure a variety of collected aspect for individual points, or
- the natural consequence of collecting a wide area of view, especially from a moving sensor.

We use MD.NomGeo.Swath.* to collect nominal values characterizing diversity of collection geometry due to the second case (swath level, as opposed to point level), above. These values are nominal in the sense that they are derived from mission planning data, coarse sensor look direction and platform data or other simple means for the purpose of characterizing the diversity of collection geometry across all positions in the collected scene. The average relative collection orientation of the sensor and target points over the full swath product populates the value of MD.NomGeo.Swath.Az.Ctr. Consider all of the single points in the vicinity of the surface of the earth where a scene to be probed by the sensor lies. Now, viewing the sensor positions from the target points over the full time period that the sensor is collecting data over the swath, conceptually build a histogram of azimuths. There is a range of azimuth portrayed by this histogram. The middle value of the azimuth axis range of this histogram spanning all bins with non-zero counts is the value populated into MD.NomGeo.Swath.Az.Ctr. As the

azimuth increases from a fraction of a degree to full circle (360 degrees), the center of the wedge in azimuth goes from clearly defined as the middle of some azimuth range (such as 10 degrees being the middle of a collection path that moved from 5 degrees to 15 degrees from North.) to problematic. When a full circle or more is traversed by the sensor as seen from the target point position, we no longer necessarily can say that half the sum of the minimum and maximum azimuths gives the center. In particular, 360 degrees is the same as zero degrees, so half of 0 and half of 360 become numerically ambiguous prescriptions. We arbitrarily define the center in azimuth of a collection that is greater than or equal to a full circle of collection to be 0 degrees. No consideration of a partial overage in the number of times the sensor orbits the target point is considered in assigning a center to the collection path in the azimuth direction.

MD.NomGeo.Swath.Az.Ctr is the center of the azimuth range from which the scene as a whole has been collected subject to the above clarifications and conventions. No averaging over points is to be performed. MD.NomGeo.Swath.Az.Ctr is effectively the mean of the lowest azimuth from the point selected from all of the points in a data set to have the lowest instantaneous azimuth, and the highest azimuth, similarly selected, over all of the points. On the other hand, MD.NomGeo.Pt.Az.Ctr is an average *over points* of the mean of the lowest and highest azimuth for each point. While this process for computing the value of MD.NomGeo.Swath.Az.Ctr is distinct from that for evaluating MD.NomGeo.Pt.Az.Ctr, it could turn out to be numerically close to the swath averaged center azimuth as defined in the paragraph for MD.NomGeo.Pt.Az.Ctr.

lidar specific note: When precise data about collection geometry for a particular point in a data set is desired, SC.Pulse.SensorPos.*TileNum*(ip,j) data should be used instead of MD.NomGeo.Swath.*.

Governing quality metrics

See Also MD.NomGeo.Pt.Az.Ctr

20.1.69 MD.NomGeo.Swath.Az.Spread [L1,2,3,4,5]

Brief Definition MD.NomGeo.Swath.Az.Spread is the range of azimuth about the central value of the collected azimuth angles over the collected swath.

indices none

Units/representation Degrees, Float

Treatment/usage MD.NomGeo.Swath.Az.Spread characterizes the swath level diversity of azimuth collection in a data set. This is a nominal value obtained by

modeling the extremes of collected azimuth angle over the swath covered in the data set consisting of individual point azimuth collection interval widths. Note that if collection is strictly limited to perfect nadir alignment with respect to the scene vertical axis, the azimuth angle of collection is undefined. In this rare case[95], the value of azimuth is reported to be -3.60E+02 degrees, and MD.NomGeo.Swath.Az.Spread is reported to be 0.0E+00. MD.NomGeo.Swath.Az.Spread supports distinguishing a scan and flight line strategy meant to gather azimuth diversity at each point in a single collection from a scan strategy and flight line choice which incidently creates some diversity across points in order to cover area. The value of MD.NomGeo.Swath.Az.Spread is just the difference in the extreme values over the product of collected azimuth when less than a full circle is collected. [as opposed to the point level azimuth collection aspect sweep variation over the points in a data set leading to an averaging operation]. So a narrow range of azimuth angles may apply to each scene point within a single scan, and re-scanning the point in the same collection operation may bring diversity of azimuth angle. But only two points contribute to the computed value of MD.NomGeo.Swath.Az.Spread. MD.NomGeo.Swath.Az.Spread considers only the lowest azimuth of the point within the collection interval that swings lowest in azimuth and the highest azimuth of the point that swings highest in azimuth over the entire data set. When a full circle or more has been collected in azimuth, the reported value of MD.NomGeo.Swath.Az.Spread should be 3.60E+02.

Governing quality metrics	
See Also	MD.NomGeo.Pt.Az.Spread

20.1.70 MD.NomGeo.Swath.Nadir.Ctr [L1,2,3,4,5]

Brief Definition	MD.NomGeo.Swath.Nadir.Ctr is the central value of the collected range of nadir angles for the full swath contained in the data set.
indices	none
Units/representation	Degrees, float
Treatment/usage	Diversity of the collection geometry in remote sensing can result from both • the intended variation of the look direction towards scene positions to assure a variety of collected aspects for individual points, or

[95] A FOV range of nadir angles over multiple detectors in an array will result in some points being collected off nadir even if the sensor is not scanning off of nadir. Only a “single pixel” device can have nadir = 0 for all collected points.

- the natural consequence of collecting a wide area of view, especially from a moving sensor.

We use MD.NomGeo.Swath.* to collect nominal values characterizing diversity of collection geometry at the swath level so that we can distinguish data collected for the first case above from that collected for the second case, and also to reveal the degree of diversity at the swath level to compare with the diversity over points in the product. These values are nominal in the sense that they are derived from mission planning data, coarse sensor look direction and platform data or other simple means for the purpose of characterizing the diversity of collection geometry at individual positions in the collected scene. These are not statistics that require quantitative assessment of each point in a data set.

Consider a single point in the vicinity of the surface of the earth where a scene to be probed by the sensor lies. Now, viewing the sensor positions from the target point over the full time period that the sensor is collecting data at the location of the point, a volume "wedge" is traced out by that part of the plane of incidence between the line of sight and the ground. There is a range of nadir angle to this wedge. As the nadir range increases from a fraction of a degree (nearly straight down viewing) to 180 degrees(looking straight up) the center of the wedge in Nadir remains clearly defined as the middle of the nadir range (such as 10 degrees being the middle of a collection path that moved from 5 degrees to 15 degrees from vertical.) as opposed to the case of azimuth, where there is the 360 degree wrap issue.

MD.NomGeo.Swath.Nadir.Ctr is the center of the nadir angle range from which all positions in the scene have been collected.

Consider all of the single points in the vicinity of the surface of the earth where a scene to be probed by the sensor lies. Now, viewing the sensor positions from the target points over the full time period that the sensor is collecting data over the swath, conceptually build a histogram of nadir angles. There is a range of nadir angles to this histogram. The middle value of the nadir axis range of this histogram spanning all bins with non-zero counts is the value populated into MD.NomGeo.Swath.Nadir.Ctr. No direct average is to be performed. To be very clear, the swath level center value of nadir angle is midway between the least nadir (most vertically downward) angle collected for any point, and the most off nadir angle collected for any point, not necessarily the same point. While this process for computing the value of MD.NomGeo.Swath.Nadir.Ctr is distinct from that for evaluating MD.NomGeo.Pt.Nadir.Ctr, it could turn out to be numerically close to the swath averaged center nadir as defined in the paragraph for MD.NomGeo.Pt.Nadir.Ctr.

Typically, if the sensor is well elevated above a scene, varying the look

direction can introduce point level nadir diversity. If the sensor is at a fixed ground level position, point level nadir diversity is not achievable, but swath level nadir diversity could be substantial depending on the field of view collected.
When precise data about collection geometry for a particular point in a data set is desired, SC.Pulse.SensorPos.*TileNum*(ip,j) data should be used instead of MD.NomGeo.Pt.*.

Governing quality metrics: Values must be populated such that the cosine of the nadir angle is within 1% of the ideal/error-free value.

See Also: MD.NomGeo.Pt.Nadir.Ctr

20.1.71 MD.NomGeo.Swath.Nadir.Spread [L1,2,3,4,5]

Brief Definition: MD.NomGeo.Swath.Nadir.Spread is the range of nadir angle about the central value of the collected nadir angles across the collected swath.

indices: none

Units/representation: Degrees, Float

Treatment/usage: MD.NomGeo.Swath.Nadir.Spread characterizes the swath level diversity of nadir angle collection in a data set. Consider all of the single points in the vicinity of the surface of the earth where a scene to be probed by the sensor lies. Now, viewing the sensor positions from the target points over the full time period that the sensor is collecting data over the swath, conceptually build a histogram of nadir angles. There is a range of nadir angles to this histogram. The difference between the maximum nadir angle and the minimum nadir angle on this axis with non-zero counts gives MD.NomGeo.Swath.Nadir.Spread. No direct average is to be performed. While this process for computing the value of MD.NomGeo.Swath.Nadir.Spread is distinct from that for evaluating MD.NomGeo.Pt.Nadir.Spread, it could turn out to be numerically close to the swath averaged spread in nadir as defined in the paragraph for MD.NomGeo.Pt.Nadir.Ctr.

MD.NomGeo.Swath.Nadir.Spread distinguishes a scan and flight line strategy meant to gather nadir diversity at each point in a single collection from a scan strategy and flight line choice which incidently creates some diversity across points in order to cover area.
To be clear, in lidar terms, MD.NomGeo.Swath.Nadir.Spread is simply the difference between the highest nadir angle over all the points (and or waveforms) in a product and the lowest nadir angle over all of the points (and or waveforms) in the product.

Governing quality metrics:

See Also MD.NomGeo.Pt.Nadir.Spread

MD.Objective.*

MD.Objective.* parameters describe the items associated with characterizing the purpose for which the data in the current data set was collected.

20.1.72 MD.Objective.Abstract [L1,2,3,4,5]

Brief Definition: MD.Objective.Abstract is a character string of human readable text describing the data set from the point of view how it is intended to meet its purpose in terms of collection and processing.

indices: none

Units/representation: Dimensionless BCS String, length PD.Wordsize.Meta(1)

Treatment/usage: MD.Objective.Abstract is a text summary of what the data set is and how it is supposed to meet its purpose. The following template gives a guideline for structuring the text that populates MD.Objective.Abstract. Items within quotation marks (“ ”) are intended to be closely replicated in the populated value of MD.Objective.Abstract. The double forward slash, “//” indicates concatenation and is not intended to be literally inserted into the MD.Objective.Abstract character string. The carriage return symbol, “C_R”, likewise acts to start a new line is not to be interpreted as printable text.

Recommended lidar Template:

“Abstract of Data Set” C_R

“Mission area:” followed by an appropriate choice from “Mapping for long term archive”, “Mapping for dynamic mission support”, “Feature/ Vertical object characterization”, “Foliage penetration”, “Tactical Surveillance” C_R

“Collection mode:” followed an appropriate choice from

“area coverage for mapping/search”, “vertical object/Folpen aspect diversity”, “Bathymetic depth”// either “sensor above surface” or “sensor below surface”, “Bathymetric obstacles”// either “sensor above surface” or “sensor below surface”,

“3-D video mode”, “System safety/security/performance study”, “Special target oriented study strategy” C_R

“Motivation for choice of time of collection start:” followed by an an appropriate choice from “target activity”, “expected viewing conditions”, “system availability” C_R

“Principal Expected Exploitation:” followed by an appropriate choice of one or more from “Point cloud analysis”, “waveform analysis”, “Assessment of derived product”, //“static”, //“video”, //“other temporal dynamic”, “generation of derived product for fusion” C_R

“Additional Comment:” followed by a free form text field

Additional items to the above suggestions can be added without compromising the validity of the population of MD.Objective.Abstract so long as the character strings used are consistent with the scope of this parameter, which is description of the objective of the data set.

This information is distinct and supplementary to MD.Objective.Desc, which is a text summary of what the primary purpose(s) for collecting the original data was.

Governing quality metrics

See Also MD.Objective.Desc PD.Wordsize.Meta(j)

20.1.73 MD.Objective.BE_NumList(j) [L1,2,3,4,5]

Brief Definition MD.Objective.BE_NumList(j) is the list of Basic Encyclopedia numbers in the objective collection requirement supported by the current data set.

indices Index j ranges from PD.LoIndx.Swath(9) to PD.HiIndx.Swath(9), that is over the number of BE numbers covered by the objective collection requirement the data set supports.

Units/representation Dimensionless, BCS string, length 15

Treatment/usage MD.Objective.BE_NumList(j) supplies a list of Basic Encyclopedia numbers that apply to the data set. Each 15 character string consists of a basic 10 character BE number with an optional 5 character extension. This extension will be post padded blanks when not applicable.

Governing quality metrics

See Also PD.HiIndx.Swath(9)

20.1.74 MD.Objective.Conops(j) [L1,2,3,4,5]

Brief Definition MD.Objective.Conops(j) provides a list of codes that describe the conops for collecting and processing the data in the current data set.

indices j=1,2

Units/representation Dimensionless, BCS string, length 15

Treatment/usage Mission planning for the collection of a lidar data set will tend to leverage the capabilities of the sensor and platform to minimize the time to reach collection objectives. The collection and processing strategies are linked to the system capabilities and the collection requirements.
MD.Objective.Conops(j) carries codes for the choices made in 2 mission areas: standalone status and timeliness. These may be used to guide data through the processing chain as well as to document the rationale for the product chain as created.

MD.Objective.Conops(1) describes the standalone status of the L-3 data sets that can be generated from the raw data.

MD.Objective.Conops(1)=1 means that the primary intent was for L-3 data sets to be exploited as stand alone data, not merged with other lidar data. L-4 generation is an option not the main plan.

MD.Objective.Conops(1)=2 means that the primary intent was for supporting derived single source data sets at L-4 that are to be exploited as stand alone data, not treated as uniquely lidar data. DEMS and tactical decision aids are examples.

MD.Objective.Conops(1)=3 that the primary intent was for void free negligibly overlapping L-3 data sets to be merged to form area mosaics at L-4 for larger coverage

MD.Objective.Conops(1)=4 that the primary intent was for L-3 data sets with planned void regions would be merged to form full coverage of the swath region at L-4 .

MD.Objective.Conops(1)=5 that the primary intent was for void free L-3 data sets to be merged to form aspect diversity at L-4 over common geographic coverage. [i.e., Signature filling at L-4]

MD.Objective.Conops(1)=6 that the primary intent was for multiple void free L-3 data sets with strong PDOP to be merged to form data sets so that geolocation errors can be reduced at L-4.

MD.Objective.Conops(1)=7 that the primary intent was for multiple L-3 data sets (at least some of which are void free) with strong mutual PDOP over a fraction of the tasked area to be processed to arrive at block adjustments for L-3 data so that geolocation errors can be reduced to L-4 levels without adding points to the basic point cloud.

Some of the above cases presume that fellow data sets are planned for co-processing. It is recommended that in those cases MD.Aggregate.Fellow.* be populated.

MD.Objective.Conops(2) describes the exploitation latency of the primary objective of collection for the data set.

MD.Objective.Conops(2)=1 means that the data set is for operational calibration and quality assessment and is to be used in as close to real time as possible for active adaptation of the collection process

MD.Objective.Conops(2)=2 means that the data set is for real time targeting and fire control or cueing of other sensors

MD.Objective.Conops(2)=3 means that the data set is for immediate delivery (several minutes latency) to exploitation stations for monitoring of operational missions. This is snapshot data, and is not persistent.

MD.Objective.Conops(2)=4 means that the data set is for immediate delivery (several minutes latency) to exploitation stations for monitoring of operational missions. This is persistent data that is continuously contributing to rolling situational awareness products. This includes lidar "video" mode.
MD.Objective.Conops(2)=5 means that the data set is for near term hours scale latency usage such as building composite/multi-Int products, tactical decision aids, or performing forensic analysis.
MD.Objective.Conops(2)=6 means that the data set is for traditional phase-2 exploitation at centers removed from the forward activity theater of operations.
MD.Objective.Conops(2)=7 means that the data set is for building foundational and strictly controlled products for long term usage at centers removed from the forward activity theater of operations.

It should be assumed that secondary usage of all data sets will occur, but the value of MD.Objective.Conops(2) should be inherited to secondary products so that the heritage can be properly understood.

Governing quality metrics

See Also: MD.Objective.Abstract MD.Aggregate.Fellow.DS_Name

20.1.75 MD.Objective.DateDue[L1,2,3,4,5]

Brief Definition: MD.Objective.DateDue is the date before which the collection was to have occurred.

indices: none

Units/representation: Days, UInt16

Treatment/usage: MD.Objective.DateDue for enterprise levels-1 through 3 is the date collection must be done before to satisfy the tasking requirements.
Following the CMMD TJD convention, this is the number of days elapsed between midnight on May 24, 1968 and the latest permitted date of the start of the collection operation for the current data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.

Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality metrics

See Also

20.1.76 MD.Objective.DateAcceptable[L1,2,3,4,5]

Brief Definition

MD.Objective.DateAcceptable is the date after which the collection was to have occurred.

indices

Units/representation

Days, UInt16

Treatment/usage

MD.Objective.DateAcceptable is the date that collection must be subsequent to. Thus, MD.Objective.DateAcceptable is an "acceptable after" criterion that opens the interval over which a collection can be counted as satisfying its tasking. The interval ends on the date due, MD.Objective.DateDue.

MD.Objective.DateAcceptable for enterprise levels-1 through 3 is the date collection must be done after to satisfy the tasking requirements.

Following the CMMD TJD convention, this is the number of days is the number of days elapsed between midnight on May 24, 1968 and the earliest permitted date of the start of the collection operation for the current data set. This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display

	as ISO 8601 compliant YYYYMMDD.
Governing quality metrics	
See Also	

20.1.77 MD.Objective.Desc[L1,2,3,4,5]

Brief Definition	MD.Objective.Desc is a human readable text describing the objective that the current data was collected and processed to meet.
indices	none
Units/representation	Dimensionless, BCS string length PD.Wordsize.Meta(2)
Treatment/usage	MD.Objective.Desc carries information describing the objective that motivated collecting the data in the current data set, e.g. "campaign to map region ABC" or a collection requirement such as “Monitor target for vehicle departure”, “Vertical object low flight safety survey of Manhattan”. This information is distinct and supplementary to MD.Objective.Abstract, which is a text summary of what the data set is and how it is supposed to meet its purpose.
Governing quality metrics	
See Also	MD.Objective.Abstract PD.Wordsize.Meta(2)

20.1.78 MD.Objective.EastBound[L1,2,3,4,5]

Brief Definition	MD.Objective.EastBound is the eastern limit of the required area that the current data set was collected to meet.
indices	none
Units/representation	Decimal degrees of longitude, Float
Treatment/usage	GeoInt collection requirements can be described in terms of regional bounds on a latitude-longitude grid. Often, the collection objectives exceed what is possible in a single collection operation. It is valuable contextual information to know what the larger intent was when deciding whether a candidate data set fulfills needs or is compatible with other data with which it is hoped to combine with to satisfy the information needs behind tasking of the collection operation(s). MD.Objective.EastBound is the eastern limit of the required area for the collection mission addressed by the current data set. It may be east of any point in the current data set since individual products may cover only part of an objective and may need to be taken together to meet a collection objective
Governing quality metrics	
See Also	MD.Objective.WestBound MD.Objective.NorthBound

20.1.79 MD.Objective.ID [L1,2,3,4,5]

Brief Definition	MD.Objective.ID is an identifier used in a tasking database for the collection requirement that is being satisfied, at least in part, by the current data set.
indices	none
Units/representation	Dimensionless, BCS string length 10
Treatment/usage	Some tasking systems may use the term "collection requirement number" or CR for the identifier of the requirement behind execution of the collection whose data is in the current data set. This field should be post padded with blanks as necessary.
Governing quality metrics	
See Also	

20.1.80 MD.Objective.NorthBound [L1,2,3,4,5]

Brief Definition	MD.Objective.NorthBound is the northern limit of the required area that the current data set was collected to meet.
indices	none
Units/representation	Decimal degrees of latitude, Float
Treatment/usage	GeoInt collection requirements can be described in terms of regional bounds on a latitude-longitude grid. Often, the collection objectives exceed what is possible in a single collection operation. It is valuable contextual information to know what the larger intent was when deciding whether a candidate data set fulfills needs or is compatible with other data with which it is hoped to combine with to satisfy the information needs behind tasking of the collection operation(s). MD.Objective.NorthBound is the northern limit of the required area for the collection mission addressed by the current data set. It may be north of any point in the current data set since individual products may cover only part of an objective and may need to be taken together to meet a collection objective
Governing quality metrics	
See Also	MD.Objective.WestBound MD.Objective.SouthBound

20.1.81 MD.Objective.SouthBound [L1,2,3,4,5]

Brief Definition	MD.Objective.SouthBound is the southern limit of the required area that the current data set was collected to meet.
indices	none
Units/representation	Decimal degrees of latitude, Float
Treatment/usage	GeoInt collection requirements can be described in terms of regional bounds on a latitude-longitude grid. Often, the collection objectives exceed what is possible in a single collection operation. It is valuable contextual information to know what the larger intent was when deciding whether a candidate data

	set fulfills needs or is compatible with other data with which it is hoped to combine with to satisfy the information needs behind tasking of the collection operation(s). MD.Objective.SouthBound is the southern limit of the required area for the collection mission addressed by the current data set. It may be south of any point in the current data set since individual products may cover only part of an objective and may need to be taken together to meet a collection objective
Governing quality metrics	
See Also	MD.Objective.WestBound MD.Objective.NorthBound

20.1.82 MD.Objective.Tasker[L1,2,3,4,5]

Brief Definition	MD.Objective.Tasker is the entitity that generated the collection requirement whose satisfaction the current data set supports.
indices	none
Units/representation	Dimensionless, BCS string length PD.Wordsize.Meta(3)
Treatment/usage	MD.Objective.Tasker is a character string carrying the name of the entity submitting the collection requirement that drove generation of the current data set. This data will be provided to the generators of CMMD compliant lidar data sets by the collecting vendor and will likely use spelling that conforms to an official registry.
Governing quality metrics	
See Also	PD.Wordsize.Meta(3)

20.1.83 MD.Objective.Vertices(iv,j) [L1,2,3,4,5]

Brief Definition	MD.Objective.Vertices(iv,j) is a list of geodetic positions which are vertices describing the required region whose satisfaction is supported by the current data set.
indices	Index, iv, ranges from PD.LoIndx.Swath(7) to PD.HiIndx.Swath(7), that is over the vertices of the bounding polygon of the region to be collected. Index, j, =1,2
Units/representation	Decimal degrees, Float
Treatment/usage	Sometimes areas of interest for collection are large and irregular in horizontal extent. In such cases, it is helpful to describe such regions in terms of the vertices of a polygon on the surface of the earth. MD.Objective.Vertices(iv,j) is a counter clockwise list of vertices beginning at the first vertex, which is the north most or the north-west vertex, which ever is unique. These vertices define the bounding polygon for the full tasked area of the collection requirement. The bounds of the current data set may or may not border on the boundary of the polygon defined here, as a collected data set may be only part of the set of collections necessary to satisfy the requirements described by

MD.Objective.Vertices(iv,j).
MD.Objective.Vertices(iv,1) carries the latitude of the ivth vertex
MD.Objective.Vertices(iv,2) carries the longitude of the ivth vertex
Latitudes and Longitudes are referenced to the WGS-84 version indicated in ES.Origin.GeodtcEpoch.

Governing quality metrics

See Also PD.HiIndx.Swath(j) ES.Origin.GeodtcEpoch

20.1.84 MD.Objective.WAC_List(j,k) [L1,2,3,4,5]

Brief Definition MD.Objective.WAC_List(j,k) is a list of **W**orld **A**eronautical **C**hart WTM sub-cells in the collection objective that the current product's collection was done to satisfy (at least in part)

indices Index, j, ranges from PD.LoIndx.Swath(8) to PD.HiIndx.Swath(8), that is, over the number of WTM subcells in the tasked region that this data set was collected to satisfy (at least in part).
Index, k, =1,2

Units/representation Decimal degrees, Float

Treatment/usage MD.Objective.WAC_List(j,k) lists the positions of the World Aeronautical Chart (WAC) Target Mosaic (WTM) sub-cells of the region in the objective collection campaign. These nominally 3x3 nautical mile cells are sub-divisions of the WTMs (24 sub-cells to the nominal 12 x 18 WTM).

MD.Objective.WAC_List(j,1) is the latitude of the jth WTM subcell centroid in the collection requirement.
MD.Objective.WAC_List(j,2) is the longitude of the jth WTM subcell centroid in the collection requirement.

By labeling the sub-cells in terms of their geodetic centroids, we make understanding of the larger collection scheme readily apparent to users without a geocoded WAC map while fully enabling a tracking system to determine satisfaction of the full requirement from lidar Product geo-referencing data and the objective collection coverage metadata. Any system that needs the WAC based names for these subcells can reconstruct them from the lat-lon data provided in this item.

Governing quality metrics

See Also MD.Coverage.Bounds.ZoneList(j) PD.HiIndx.Swath(j).

20.1.85 MD.Objective.WBStyle [L1,2,3,4,5]

Brief Definition: MD.Objective.WBStyle indicates form of the sensor or sensor[96]derived data in the current data set.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: MD.Objective.WBStyle is the wide band data style of the data in the current data set. Typically remote sensing data sets and data sets derived from remote sensing systems have two parts: the narrowband *meta*data and the wideband sensor data or data derived from it. The metadata section is typically a much smaller byte volume percentage (thus "narrowband") of a data set than is the sensor measurement based data, which often exceeds millions (thus "wideband ") of samples of sensed data or consequences of processing that data. While not impossible, the probablility that the descriptive data is more voluminous than the sensor measurement based data is small in customary data set preparations. So MD.Objective.WBStyle is a reference to the style of the sensor measurement based data set content as opposed to the metadata.

There are processing method options (i.e., wideband "strategies") that lead to a variety of ways the samples in the wide band data represent the measurements. These strategies are generally tied to the enterprise product level of the data set.

MD.Objective.WBStyle is a 3 digit code that can be applied to describe the wideband strategy of the data in a data set.
Codes 000 through 127 are for active sensors such as lidar and Radar. Codes 128 through 255 are for passive systems such as E-O and acoustic systems.

Style code value	Sensor type(s)	Data strategy
000	Active Optical (lidar)	Raw proportional to Rx amplitude (~volts)
001	Active Optical (lidar)	Raw proportional to Rx intensity (~amps in photo current)
002	Active Optical (lidar)	Raw Proportional to Rx fast time (~clock counts)
003	Active Optical (lidar)	Coded Raw Rx amplitude (e.g. when VQ codes used)
004	Active Optical	Coded Raw Rx intensity

[96] Text written to facilitate use as a common metadata item. For a lidar document, this generalized reference to a "sensor" would be interpreted as a reference to a lidar sensor.

	(lidar)	
005	Active Optical (lidar)	Processed to fast time clips (aka waveforms) only
006	Active Optical (lidar)	Processed to scene positions in sensor frame only
007	Active Optical (lidar)	Processed to scene positions in earth frame only
008	Active Optical (lidar)	Processed to scene positions in sensor frame and fast time clips (aka waveforms)
009	Active Optical (lidar)	Processed to scene positions in earth frame and fast time clips (aka waveforms)
010	Active Optical (lidar)	Processed to scene positions with intensities in sensor frame only
011	Active Optical (lidar)	Processed to scene positions with intensities in earth frame only
012	Active Optical (lidar)	Processed to scene positions with intensities in sensor frame and fast time clips (aka waveforms)
013	Active Optical (lidar)	Processed to scene positions with intensities in earth frame and fast time clips (aka waveforms)
014	Active Optical (lidar)	Reserved
015--020	Active Optical (lidar)	Reserved[97]
…[98]		

The items in this table are more comprehensive than the stages listed for the product types and should be considered supplementary information to MD.ProcHist.ProdType.

Governing quality metrics

See Also MD.ProcHist.ProdType

20.1.86 MD.Objective.WestBound [L1,2,3,4,5]

Brief Definition MD.Objective.WestBound is the western limit of the required area that the current data set was collected to meet.

[97] Might reserve for derivative product types such as TINs, DEMS of various kinds ("Bare Earth" 1, Bare Earth 2, Surface 1, Surface 2,… when issues become better resolved regarding definitions regarding voids, synthetic voids, first and last return, features, noise, dynamic scene content , levels of hydrologic control, etc. The list should be registered so it can be addressed by documents for non-lidar data such as DEMs.

[98] Other sensor and passive sensor data get rows too, eventually. Must decide about reuse of options introduced earlier in the table and cross- sensor options. For example does radar amplitude proportional data get a seprate code, or do we just put Radar into the second column of row/code 000? This is a common metadata CM issue.

indices	none
Units/representation	Decimal degrees of longitude, Float
Treatment/usage	GeoInt collection requirements can be described in terms of regional bounds on a latitude-longitude grid. Often, the collection objectives exceed what is possible in a single collection operation. It is valuable contextual information to know what the larger intent was when deciding whether a candidate data set fulfills needs or is compatible with other data with which it is hoped to combine with to satisfy the information needs behind tasking of the collection operation(s). MD.Objective.WestBound is the western limit of the required area for the collection mission addressed by the current data set. It may be west of any point in the current data set since individual products may cover only part of an objective and may need to be taken together to meet a collection objective
Governing quality metrics	
See Also	MD.Objective.EastBound MD.Objective.NorthBound

MD.Platform.*

MD.Platform.* parameters describe the items associated with characterizing the collection platform.

20.1.87 MD.Platform.Desc(j) [L1,2,3,4,5]

Brief Definition	MD.Platform.Desc(j) is a readable text description of the platform(s) carrying the sensor(s) used to collect the current data set.
indices	Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of collections contributing data to the current data set.
Units/representation	Dimensionless, BCS String length 20
Treatment/usage	MD.Platform.Desc(j) carries free text descriptions of the sensor platform(s), e.g ."Int'l Space Station", or "prop plane", etc as an adjunct to the information in MD.Platform.ID(j). For enterprise level-2 data sets, PD.LoIndx.Parents will always equal PD.HiIndx.Parents , so only one platform gets a description. This field will be post padded with blanks as appropriate.
Governing quality metrics	
See Also	MD.Platform.ID(j)

20.1.88 MD.Platform.ID(j) [L1,2,3,4,5]

Brief Definition	MD.Platform.ID(j) is a list of identifiers for the platform(s) carrying the sensor(s) used to collect the current data set.
indices	Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over

	the number of collections contributing data to the current data set. For enterprise L-1, L-2, and L-3 data sets, only one sensor may contribute to a data set. Only enterprise L-4 data sets may have more than a single sensor as data sources.
Units/representation	Dimensionless, character string length 6
Treatment/usage	MD.Platform.ID(j) is a list of registered 6 character identifiers specific to the platforms bearing the sensors used to create the current data set. These identifiers would be, for example a tail number, Satellite ID (e.g. for IceSat or Calipso), or a pedestal number in the case of ground fixed platforms. Note that the indexing is over collections. If the same platform is used in more than one collection contributing to the current data set, the value of the platform ID will be repeated for more than one value of index, j. This is the searchable parameter that is the same as the platform ID sub-string of MD.Meta.DS_Name. The length of this character string has been chosen to support using US N-Numbers up to N999ZZ as platform identifiers.
Governing quality metrics	
See Also	MD.Sensor.ID(j) MD.Meta.DS_Name

MD.ProcHist.*

MD.ProcHist.* parameters describe the items associated with characterizing the processing history of the data in the current data set.

20.1.89 MD.ProcHist.CodeList(j) [L1,2,3,4,5]

Brief Definition	MD.ProcHist.CodeList(j) is a list of product type descriptions that are referenced to describe the current product and its precursors.
indices	Index, j, ranges from PD.LoIndx.MD_Tree(3) to PD.HiIndx.MD_Tree(3), that is over the distinct product type codes referenced in the current data set.
Units/representation	Dimensionless, BCS Character string(33)
Treatment/usage	This list is defined to simplify the structure of the MD.Aggregate.ProdTypeTree(i,j,k) array and serve MD.Aggregate.Fellow.ProdType(j) and MD.ProcHist.ProdType ,which also reference strings that describe product types in terms of the processing heritage of the data sets. The value of index, j, corresponding to a particular string value in MD.ProcHist.CodeList(j) acts as the code for the string. Thus, any reference to a ‘product type’ code is resolved to a character string value by a lookup in this table.
Governing quality metrics	
See Also	PD.HiIndx.MD_Tree(3) MD.ProcHist.ProdType

20.1.90 MD.ProcHist.L1Writer.Name(k) [L1,2,3,4]

Brief Definition	MD.ProcHist.L1Writer.Name(k) carries the name for the application that created the enterprise L1 data set(s) leading to the current data set.
indices	Index k ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, and 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Dimensionless BCS string length 20
Treatment/usage	This field may be populated for data sets at level-1 and above. The enterprise level-1 writer application name refers to the name of the piece of software used to create the enterprise level-1 data set(s) at the root of the PED for the current product. This will encode a description of the software function or its vendor's way of referencing the application, but shall not encode the release date, vendor, or version of the level-1 writing application as these are all directly supported by separate metadata items. This field will be post padded with blanks as appropriate. A possible value might be, "GMlidarL1Transr "
Governing quality metrics	
See Also	PD.HiIndx.Parents MD.ProcHist.L1Writer.Versn(k) MD.ProcHist.L2App.Name(j,k)

20.1.91 MD.ProcHist.L1Writer.ProcDate(k) [L1,2,3,4]

Brief Definition	MD.ProcHist.L1Writer.ProcDate(k) gives the creation date for the enterprise level-1 data set(s) used for the current data set.
indices	Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, and 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Days, UInt16
Treatment/usage	MD.ProcHist.L1Writer.ProcDate(k) supplies a list of dates on which the enterprise level-1 contributors to the current data set were generated. Dates will be represented in this metadata entity as TJD and are convertible to YYYYMMDD for display purposes. MD.ProcHist.L1Writer.ProcDate(k) is the number of days elapsed between

midnight on May 24, 1968 and the date of the start of L1 generation for the kth level-1 used to create the current data set.
This value is computed per NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality metrics

See Also: MD.Collect.CollctnDate

20.1.92 MD.ProcHist.L1Writer.ProcSite(k) [L1,2,3,4]

Brief Definition: MD.ProcHist.L1Writer.ProcSite(k) gives the processing site for the enterprise level-1 data set(s) used for the current data set.

indices: Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.

Units/representation: Dimensionless, BCS length 20

Treatment/usage: The character string(s) in MD.ProcHist.L1Writer.ProcSite(k) indicate where the enterprise level-1 data set(s) contributing to the current data set was/were generated. Production facilities are free to use a registered list of spellings for such processing sites that can be used to assure recognition of the values populated into MD.ProcHist.L1Writer.ProcSite(k). However, since this data is informational and likely to be processed only by human readers, such regulation of the values used to populate this field may not be deemed necessary.

Governing quality metrics
See Also

20.1.93 MD.ProcHist.L1Writer.RelDate(k) [L1,2,3,4]

Brief Definition
MD.ProcHist.L1Writer.RelDate(k) gives the writer application release date for the enterprise level-1 data set(s) used for the current data set.

indices
Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.

Units/representation
Days, UInt16

Treatment/usage
MD.ProcHist.L1Writer.RelDate(k) carries a list of release dates, one per enterprise L1 writer, that indicate when the versions of the software used to create the contributing enterprise level-1 data sets for the current data set were published. All values in MD.ProcHist.L1Writer.RelDate(k) are Truncated Julian Day and convertible toYYYYMMDD for display. Each value is the number of days elapsed between midnight on May 24, 1968 and the date of the release of the L1 writer used in the processing chain for the current data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality metrics
See Also

20.1.94 MD.ProcHist.L1Writer.Vendor(k) [L1,2,3]

Brief Definition	MD.ProcHist.L1Writer.Vendor(k) gives the writer application vendor for the enterprise level-1 data set(s) used for the current data set.
indices	Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Dimensionless. BCS string 20
Treatment/usage	MD.ProcHist.L1Writer.Vendor(k) supplies a list of the names of the vendors providing the application for writing each L1 data set contributing to the current data set. Abbreviations may be used to assure that the vendor name is unambiguously represented with the 20 character limit and blanks will be post padded to fill any unused space in the vendor name field. Production facilities are free to use a registered list of spellings for such vendor names that can be used to assure recognition of the values populated into MD.ProcHist.L1Writer.Vendor(k). However, since this data is informational and likely to be processed only by human readers, such regulation of the values used to populate this field may not be deemed necessary.
Governing quality metrics	
See Also	

20.1.95 MD.ProcHist.L1Writer.Versn(k) [L1,2,3,4]

Brief Definition	MD.ProcHist.L1Writer.Versn(k) gives the writer application version for the enterprise level-1 data set(s) used for the current data set.
indices	Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Dimensionless, BCS string length 5
Treatment/usage	The enterprise level-1 S/W Version Number will be of the form, XX.XX. For example, “01.09”. Each level-1 writer involved in creating the current data set will have its own version number supplied.

Governing quality metrics

See Also MD.ProcHist.L1Writer.Name(k)

20.1.96 MD.ProcHist.L2App.Name(j,k) [L2,3,4]

Brief Definition MD.ProcHist.L2App.Name(j,k) gives the application name for generating the enterprise level-2 data set(s) used for the current data set.

indices Index, j, ranges from 1 to Max[99][MD.ProcHist.L2NumStages(k)], that is, we provide an application name for each stage, j, of level-2 processing provided by a separate application leading to the current data set.
Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.

Units/representation Dimensionless BCS string length 20

Treatment/usage The enterprise level-2 Application Name refers to the name of the piece of software used to create the enterprise level-2 data set(s) used to form the current product. This will encode a description of the software function or its vendor's way of referencing the application, but shall not encode the release date, vendor, or version of the level-2 generation application as these are all directly supported by separate metadata items.
This field will be post padded with blanks as appropriate. A possible value might be, "LML2Pts&ClipsGen " (**L**inear **M**ode **L**evel-**2**)
This field may be populated for data sets at level-2 and above. Each parent file, k, of the current data set has a list of names for the level-2 generating apps used in generating the inputs to the current data set. The j stages generated using separate applications have their own names and are given here. If not all parents to the current data set have the same number of level-2 generating stages, then unused name slots are to be filled with blanks.

Governing quality metrics

See Also MD.ProcHist.L1Writer.Name(k) MD.ProcHist.L2NumStages(k)

[99] That is, over the k values of the number of stages, considering all parents in general. Note that for level-2 there is only one parent for a data set.

20.1.97 MD.ProcHist.L2App.ProcDate(j,k) [L2,3,4]

Brief Definition

MD.ProcHist.L2App.ProcDate(j,k) gives the processing date for the enterprise level-2 data set(s) used in generation of the current data set.

indices

Index, j, ranges from 1 to Max[99][MD.ProcHist.L2NumStages(k)], that is, we provide a processing date for each stage, j, of processing provided by a separate application leading to the current data set.

Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products. Retaining it in the data definition permits common metadata use across enterprise levels for this item.

Units/representation

Days, UInt16

Treatment/usage

MD.ProcHist.L2App.ProcDate(j,k) supplies a list of dates on which the enterprise level-2 contributors to the current data set were generated.

Dates will be represented in this metadata entity as TJD and are convertible to YYYYMMDD for display purposes.

MD.ProcHist.L2App.ProcDate(j,k) is the number of days elapsed between midnight on May 24, 1968 and the date of the start of a level-2 generation event for the current data set.

This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via

TJD = INT[JD − 2440000.5]

A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.

Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Each parent file, k, of the current data set has a list of processing dates for the level-2 generating apps used in generating the inputs to the current data set. The j level-2 stages generated using separate applications have their own

processing dates and are given here.

Governing quality metrics

See Also

20.1.98 MD.ProcHist.L2App.ProcSite(j,k) [L2,3,4]

Brief Definition: MD.ProcHist.L2App.ProcSite(j,k) gives the processing site(s) for the enterprise level-2 data set(s) used for the current data set.

indices: Index, j, ranges from 1 to Max[99][MD.ProcHist.L2NumStages(k)], that is, we provide a processing site for each stage, j, of processing provided by a separate application leading to the current data set.

Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.

Units/representation: Dimensionless, BCS length 20

Treatment/usage: The character string(s) in MD.ProcHist.L2App.ProcSite(j,k) indicate where the enterprise level-2 data set(s) contributing to the current data set was/were generated. Production facilities are free to use a registered list of spellings for such processing sites that can be used to assure recognition of the values populated into MD.ProcHist.L2App.ProcSite(j,k). However, since this data is informational and likely to be processed only by human readers, such regulation of the values used to populate this field may not be deemed necessary.

Each parent file, k, of the current data set has a list of processing sites for the level-2 generating apps used in generating the inputs to the current data set. The j level-2 stages generated using separate applications have their own processing sites and are given here. If not all parents to the current data set have the same number of level-2 generating stages, then unused site slots are to be filled with blanks

Governing quality metrics

See Also

20.1.99 MD.ProcHist.L2App.RelDate(j,k) [L2,3]

Brief Definition: MD.ProcHist.L2App.RelDate(j,k) gives the application release date for the enterprise level-2 data set(s) used for the current data set.

indices: Index, j, ranges from 1 to Max[99][MD.ProcHist.L2NumStages(k)], that is, we provide a release date for each stage, j, of processing provided by a separate

application leading to the current data set.
Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is, over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.

Units/representation: Days, UInt16

Treatment/usage: MD.ProcHist.L2App.RelDate(j,k) carries a list of release dates, one per enterprise L2 application, that indicate when the versions of the software used to create the contributing enterprise level-2 data sets for the current data set were published. All values in MD.ProcHist.L2App.RelDate(j,k) are Truncated Julian Day and convertible toYYYYMMDD for display. Each value is the number of days elapsed between midnight on May 24, 1968 and the date of release of the application version used to create the L2 for the current data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD[87]) is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Each parent file, k, of the current data set has a list of release dates for the level-2 generating applications used in generating the inputs to the current data set. The j level-2 stages generated using separate applications have their own release dates and are given here. If not all parents to the current data set have the same number of level-2 generating stages, then unused date slots are to be filled with 0 values, which will always predate the time period of data generation.

Governing quality metrics
See Also

20.1.100 MD.ProcHist.L2App.Vendor(j,k) [L2,3,4]

Brief Definition	MD.ProcHist.L2App.Vendor(j,k) gives the application vendor name for the enterprise level-2 data set(s) used for the current data set.
indices	Index, j, ranges from 1 to Max[99][MD.ProcHist.L2NumStages(k)], that is, we provide an application vendor name for each stage, j, of processing provided by a separate application leading to the current data set. Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Dimensionless. BCS string 20
Treatment/usage	MD.ProcHist.L2App.Vendor(j,k) supplies a list of the names of the vendors providing the application for generating each L2 data set contributing to the current data set. Abbreviations may be used to assure that the vendor name is unambiguously represented with the 20 character limit and blanks will be post padded to fill any unused space in the vendor name field. Production facilities are free to use a registered list of spellings for such vendor names that can be used to assure recognition of the values populated into MD.ProcHist.L2App.Vendor(j,k). However, since this data is informational and likely to be processed only by human readers, such regulation of the values used to populate this field may not be deemed necessary. This field may be populated for data sets at level-2 and above. Each parent file, k, of the current data set has a list of names for the level-2 generating app vendors used in generating the inputs to the current data set. The j stages generated using separate applications have their own vendor names and are given here. If not all parents to the current data set have the same number of level-2 generating stages, then unused name slots are to be filled with blanks.
Governing quality metrics	
See Also	

20.1.101 MD.ProcHist.L2App.Versn(j,k) [L2,3,4]

Brief Definition	MD.ProcHist.L2App.Versn(j,k) gives the application version identifier for the enterprise level-2 data set(s) used for the current data set.
indices	Index, j, ranges from 1 to Max[99][MD.ProcHist.L2NumStages(k)], that is, we provide an application verson number for each stage, j, of processing provided by a separate application leading to the current data set. Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Dimensionless, BCS string length 5
Treatment/usage	Enterprise level-2 software Version Numbers will be of the form, XX.XX. For example, “01.09”. Each level-2 generating application involved in creating the current data set will have its own version number supplied. This field may be populated for data sets at level-2 and above. Each parent file, k, of the current data set has a list of versions for the level-2 generating apps used in generating the inputs to the current data set. The j stages generated using separate applications have their own versions and are given here. If not all parents to the current data set have the same number of level-2 generating stages, then unused version slots are to be filled with blanks.
Governing quality metrics	
See Also	

20.1.102 MD.ProcHist.L2NumStages(k)[L2,3,4]

Brief Definition	MD.ProcHist.L2NumStages(k) gives the number of level-2 processing stages for the enterprise level-2 data set(s) used to create the current data set.
indices	Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products.
Units/representation	Dimensionless, UInt8
Treatment/usage	For the kth parent data set, MD.ProcHist.L2NumStages(k) is the number of L-2 stages that were created with separate applications executed in making the current data set. This accounting of stages increments each time an

application is necessary to advance a stage of processing. Thus, if two stages of the data are technically generated internal to a single application, the stage count is incremented by 1, not two. The intent is to track the applications used to get to the current data. The descriptions of those applications can be consulted to understand what they did technically.

Governing quality metrics

See Also

20.1.103 MD.ProcHist.ProdType(j) [L1,2,3]

Brief Definition

MD.ProcHist.ProdType(j) indicates the enterprise product level and processing stage of the stored products in the thread ending in the current dataset. This is a discovery level summary of the processing objectives performed to create the current data set.

indices

Index, j, ranges from PD.LoIndx.PrevStages(k) to PD.HiIndx.PrevStages(k), that is over the previously stored datasets processed in a chain to get to the current dataset. Since this is restricted to enterprise levels 1, 2 and 3, there is only one value of k and thus one range of index, j. When the value of j equals PD.HiIndx.PrevStages(k), MD.ProcHist.ProdType(j) gives information for the current product. When the value of j equals PD.LoIndx.PrevStages(k), MD.ProcHist.ProdType(j) gives information for the first dataset in the PED processing chain.

Units/representation

Dimensionless, UInt8

Treatment/usage

Common metadata section

MD.ProcHist.ProdType(j) is a code for a descriptive character string, which is determined by lookup in MD.ProcHistCodeList(j).
MD.ProcHist.CodeList(j) is a facilitating table providing character strings for all product types referenced in the current dataset. But to determine the product type of the current data set or a precursor, one refers to the value of the codes populating MD.ProcHist.ProdType(j). Consequently we will provide the meaning of the codes in this definition paragraph.

We describe here the product type description character strings listed in MD.ProcHistCodeList(j). These should be post-padded with blanks as appropriate. The first part of the character string is "L*n*.", where *n* has the value 1, 2, or 3, depending on the enterprise level of the jth dataset described by MD.ProcHist.ProdType(j).
The 2 character enterprise level indicator and delimiting "." are followed by a sequence of product form and processing codes of the form "S*nxy*"[100], also dot delimited. Here *n* may take the value 1,2,3, 4 etc, and *x* and *y* take the values a,b,c.
Level codes [L*n*]:
"L1", "L2", "L3" Note that these are the last two characters of the MD.Meta.DS_Name sub-field referred to as "Prod type".

Using the above codes in combination with sensor specific codes {Sensor codes listed in addendum below}, we create strings such as "L2.S1bb.S2a.S2b1" to populate MD.ProcHist.ProdType(j). These strings give a quick overview of the full processing chain invoked to get to a current

[100] "S" for stage or style.

Common metadata section

product. This is more information than available in the MD.Objective.WBStyle parameter, which contains a less specific version of only the last dot delimited field in MD.ProcHist.ProdType(j).
Also note that the codes defined for MD.ProcHist.ProdType are intended to provide more detail than the two character level-n stage/Form sub-field of MD.Meta.DS_Name. The MD.Meta.DS_Name content is high level to act only as a cue for users managing files in an exploitation environment while MD.ProcHist.ProdType(j) is intended to be a searchable, more informative description. The sub-fields are cumulative and the longer string can convey more information than the dataset name is intended to carry. There is clearly a correspondence, however, between the code lists for MD.ProcHist.ProdType(j) and the data set name subfield for stage/form.

This document does not govern any derived products. Thus, such products as 'single collect, TIN with features', 'single collect, TIN Bare earth', 'single collect, bare earth DEM', and 'Dictionary based Feature map' are not given processing stage labels here.
For datasets at product level-2 or -3, the string, "L*n*." will appear once for each value of the index, j. This product level designator labels thye current product and its precursors even in the case that there are multiple level-2 or -3 precursors to the current dataset. The "S" strings will indicate the processing history within any precursor products.

<u>lidar specific addendum</u>

Product form and processing stage codes [S*nxy*]:
"S1a" -- Sensor frame XYZ points
"S1aa" -- Sensor frame XYZ points + Waveforms
"S1b" -- Earth frame XYZ points
"S1bb"-- Earth frame XYZ points + Waveforms
"S2a" -- (denoised by coincidence processing)
"S2aa" -- (denoised by Fourier Filtering, variable threshold)
"S2bb" -- (denoised by Fourier Filtering, single set threshold value)
"S2b1"-- scans have been constructed using pulse to pulse registration
"S2b2"-- subswaths have been created using scan to scan registration[101]
"S2b3"-- the full swath has been created using subswath to subswath

[101] If the need arises to create a L2 product where subswaths have not been composed into a full swath by registration, a code can be invented to describe that state. In this version of the Lidar CMMD, it is assumed that all saved L2 data would be either prior to scan to scan registration or after subswath to subswath registration.

Registration
"S2c" -- Indicator for a fellow stage 2 product type where the fellow participates in geometric adjustment determination with the current dataset
"S3a" -- L3 cartesian frame data revised for improved accuracy based on geometrically diverse multilook data, but no wide-band merger has been performed.
"S3b" -- Single source point clouds with lidar points truth tagged [102]

Note that the L*n*, S1, and S2a codes are internally exclusive (e.g. data can't be both earth frame points and sensor frame points)

More specific information at the exploitation support level may be provided via the user defined data descriptions in PD.UserDefined.NB_Def.P*n* and PD.UserDefined.PtDef.P*n*.

Governing quality metrics

See Also PD.HiIndx.PrevStages(k) MD.Meta.DS_Name PD.UserDefined.PtDef.P*n* MD.Objective.WBStyle MD.ProcHist.CodeList(j) PD.UserDefined.NB_Def.P*n*

MD.Qual.*

MD.Qual.* parameters describe the items associated with characterizing the quality and accuracy of the data in the current data set.

20.1.104 MD.Qual.GeoLocUncH [L1,2,3,4,5]

Brief Definition: MD.Qual.GeoLocUncH is the nominal horizontal extent of the geolocation error volume for the product.

indices: none

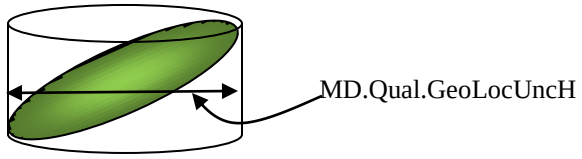
Units/representation: Meters, Float

Treatment/usage: MD.Qual.GeoLocUncH gives a characteristic value for the full data set of the horizontal uncertainty of point locations. The term, "horizontal" is used in this definition to indicate a direction parallel to the WGS-84 ellipsoid tangent plane in the vicinity of any data sample. In the rare situation that a product covers so large a region that a single horizontal direction does not apply acceptably[103] accurately to all samples in the product,

[102] The items in gray are typically associated with enterprise level-3 datasets, but provided here for context and flexibility.

[103] A calculation of characteristic horizontal or vertical error must account for the magnitude of the error. While this is a nominal value for discovery and not for, or due to, rigorous error propagation, the values should be

MD.Qual.GeoLocUncH will be an average over the local horizontal uncertainties.
MD.Qual.GeoLocUncH is the length in meters of a horizontal line through a vertical (relative to the WGS-84 ellipsoid) cylinder tangent to the 90% error ellipsoid volume. This is typically a larger measure than given by a horizontal line through the envelope of the error ellipsoid, which does not account for the impact of vertical uncertainty on the horizontal uncertainty.



To form MD.Qual.GeoLocUncH, the "90% ellipsoid" used should enclose the region of local 90% error volumes for 90% of the points in the product.

Governing quality metrics

See Also

20.1.105 MD.Qual.GeoLocUncV[L1,2,3,4,5]

Brief Definition: MD.Qual.GeoLocUncV is the nominal vertical extent of the geolocation error volume for the product.

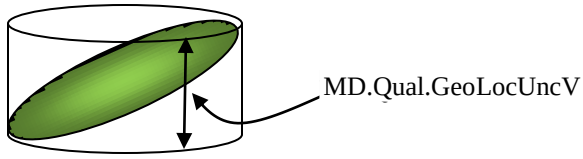
indices: none

Units/representation: Meters, Float

Treatment/usage: MD.Qual.GeoLocUncV gives a characteristic value for the full data set of the vertical uncertainty of point locations. The term, "vertical" is used in this definition to indicate a direction normal to the WGS-84 ellipsoid tangent plane in the vicinity of any data sample. In the rare situation that a product covers so large a region that a single vertical direction does not apply acceptably[103] accurately to all samples in the product,
MD.Qual.GeoLocUncV will be an average over the local vertical uncertainties.
MD.Qual.GeoLocUncV is the length in meters of a vertical line through a vertical (relative to the WGS-84 ellipsoid) cylinder tangent to the 90% error ellipsoid volume. This is typically a larger measure than given by a vertical

representative. Very low uncertainties may require more elaborate correction for changes in the horizontal direction over a product. In practical cases this will not be an issue for the forseeable future.

line through the envelope of the error ellipsoid, which does not account for the impact of horizontal uncertainty on the vertical uncertainty.



To form MD.Qual.GeoLocUncV, the "90% ellipsoid" used should enclose the region of local 90% error volumes for 90% of the points in the product.

Governing quality metrics

See Also MD.Qual.GeoLocUncH

20.1.106 MD.Qual.GSD [L1,2,3,4,5]

Brief Definition MD.Qual.GSD, the characteristic ground sample distance for a data set, gives a representative value over the product of the geometric mean of sensor pixel major and minor axis lengths as projected to the ground.

indices none

Units/representation Meters, float

Treatment/usage Each detector element in an imaging sensor has a cone of individual field of regard. Typically, imaging optical systems are operated so that they are not diffraction limited, that is, the field of regard of a detector element is large compared to the diffraction limited field of view. In this case, the projection of a sensor pixel to the ground is generally asymmetric, being an ellipse or rectangle with major and minor axes. MD.Qual.GSD is the geometric mean of the major and minor axes of this projected pixel; in the case of detector arrays with many elements, these axes may or may not be equivalent to element to element angular separation in row and column directions as projected to the ground.

Note, the ground sample distance is not a measure of how closely packed points in a product are in the earth frame. For that, see MD.Qual.RetPtDensity.

The 'representative value over the product' shall be conveyed by populating MD.Qual.GSD with a value that is at the mean plus one standard deviation in the presence of GSD variation. As this is a characterization, this value may be determined from a model of the collection process and need not carry the exactness of a detailed statistical analysis of the data collected.

Governing quality metrics

See Also MD.Qual.RetPtDensity ES.Swath.PSF(j)

20.1.107 MD.Qual.RetPtDensity[L1,2,3,4,5]

Brief Definition	MD.Qual.RetPtDensity is a point cloud quality metric that indicates the peak number of points that could be observed in an active cubic meter scattering volume of the current product. This value is not derated due to point clouds being mostly empty.
indices	none
Units/representation	Returns/meter^3, Float
Treatment/usage	MD.Qual.RetPtDensity applies to 3-D imagery, point clouds in particular. MD.Qual.RetPtDensity is the mean number of returns reportable per cubic meter over the data set. In a sense, this would indicate how many leaf returns could be seen in a volume. This quantity is determined by the voxel volume in cubic meters, ***v***, and interrogations, ***I***, (i.e. pulses times potential returns[104] per pulse) per voxel. MD.Qual.RetPtDensity (**p**otential **r**eturn **p**oint **d**ensity , or PRPD for short) is the value characteristic over the whole product, given by PRPD=***I/v***. (Note that we define the reported PRPD metric such that ***v*** =1 meter^3.) Generally most cubic meter volumes will have far fewer points than this figure. The average cubic meter of a point cloud would have[105] point counts of order **I**/ MD.Qual.GSD^2 for voxels near a reflective 2-d surface like that of the earth. Note that this quantity will include the impact of any decimation performed in generating the current product.
Governing quality metrics	
See Also	MD.Qual.GSD

[104] "Potential returns" counts all the ranges and sensing "rays" for a pulse that could be giving separately resolved returns. In single quench Geiger mode, the potential return is only one per ray. In Linear family modes, the number of range resolution cells within the meter voxel is the potential return count per ray.

[105] Assuming a mode/design strategy where the pixel separation is about the same as the GSD.

20.1.108 MD.Qual.Impairments(j) [L2,3,4,5]

Common metadata section

Brief Definition: MD.Qual.Impairments(j) reports codes for a set of degradation categories.
indices: Index j=1 through 6
Units/representation: Dimensionless, UInt8
Treatment/usage: MD.Qual.Impairments(j) is a generic list of quality impairment codes defined as:
MD.Qual.Impairments(1)—native noise levels and processing noise residuals
MD.Qual.Impairments(2)---noise processing impact to scene component detectability
MD.Qual.Impairments(3)---noise impact to edges in the scene
MD.Qual.Impairments(4)---artifacts and spurious responses
MD.Qual.Impairments(5)---geometric distortion
MD.Qual.Impairments(6)---periodic discontinuities
MD.Qual.Impairments(7)---micro-voids.

This item characterizes the prevalence of micro-voids, which are small localized regions of no return that are too numerous and irregular to be captured in MD.Coverage.Voids.*. These are dominantly dependent on the sensor response to the scene, such as the absence of retro-reflection for a monostatic lidar.

Noise is heavy when the utility of the data set is strongly degraded due to the noise.
Noise is moderate when the noise is highly noticeable, but much of the expected utility of the data can be salvaged.
Noise is light when it is unnoticeable or, if visible, not limiting to the use of the data set.

Sensor specific code value definitions follow:

<u>lidar Addendum</u>

MD.Qual.Impairments(1)

code=1 Noise	heavy untreated
code=2 Noise	heavy treated, low residual
code=3 Noise	heavy treated, moderate residual
code=4 Noise	heavy treated, high residual
code=5 Noise	moderate untreated
code=6 Noise	moderate treated, low residual
code=7 Noise	moderate treated, moderate residual

code=8 Noise light untreated
code=9 Noise light treated, low residual

MD.Qual.Impairments(2)
code =1 Noise treatment known to suppress small compact objects only
Code=2 Noise treatment known to suppress 1-D objects only
Code=3 Noise treatment known to suppress 1-D objects and small compact objects
Code=4 Noise treatment known to suppress non-zero slope surfaces
Code=5 Noise treatment known to suppress slope surfaces oblique to lidar beam
Code=6 small base vertical objects suppressed due to nadir collection and noise treatment.

MD.Qual.Impairments(3)
Code=1 Noise treatment known to preserve edge integrity
Code=2 Noise treatment known to corrupt edge integrity

MD.Qual.Impairments(4)
Code=1 lidar has systematic artifacts that are not treated: range ghost
Code=2 lidar has systematic artifacts that are not treated: blooming Horizontal only
Code=3 lidar has systematic artifacts that are not treated: blooming horizontal skirt with range smear only
Code=4 Combination of code 1 and code 2 effects
Code=5 Combination of code 1 and code 3 effects
Code=6 lidar has systematic artifacts that are treated: range ghost
Code=7 lidar has systematic artifacts that are treated: blooming Horizontal only
Code=8 lidar has systematic artifacts that are treated: blooming horizontal skirt with range smear only
Code=9 Combination of code 6 and code 7 effects
Code=10 Combination of code 6 and code 8 effects

MD.Qual.Impairments(5)
Code=0 No geometric distortion (relative errors are all random at the part in a thousand level)
Code =1 Geometric distortion levels (correlated relative errors) exceed 0.01 = stretch or bending rate for line-like objects

MD.Qual.Impairments(6)

Code=0 No noticeable "corn rowing" due to periodic z shears
Code=1 Cornrowing visible due to reporting precision, but within spec.
Code=2 noticeable but limited regions of out of spec periodic z shears
Code=3 Pervasive regions of out of spec periodic z shears
Code=4 noticeable regions of in spec periodic horizontal shears
Code=5 noticeable but limited regions of out of spec periodic horizontal shears
Code=6 pervasive regions of out of spec periodic horizontal shears

MD.Qual.Impairments(7)

Primary causes for lidar are: low reflectivity scattering (especially pooled water for lidars at wavelengths that are non-responsive to water covered regions), or oblique specular reflectors in the scene.

Code= negative number

indicates that this metric has not been evaluated.

Code= positive number.

This is the percentage (values 0 to 100) of tiles in the data set that have a 2-D horizontal point density 50% or less than the nominal 2-D reflection point density found from computing $(\text{MD.Qual.RetPtDensity})^{2/3} \times \cos(\text{nadir angle})$. This formula needs to be evaluated by individual nadir angle when there is aspect diversity.

Governing quality metrics

See Also

20.1.109 MD.Qual.XrangeAvDia(j) [L2,3,4,5]

Brief Definition: MD.Qual.XrangeAvDia(j) gives the least and greatest values of the effective cross range averaging circle diameter over the product.

indices: Index j= 1,2 (or 0,1 depending on the convention of the implementing software)

Units/representation: Meters, single float

Treatment/usage: Algorithms for cleaning up lidar point clouds by creating points confined to hypothesized surface positions typically employ cross range averaging as part of the surface range estimation process. The actual area averaged may be of some arbitrary shape, but we characterize its linear size by the diameter of a circle with the same area as that averaged. The circle is computed as if there is no weighting in the average although the actual average performed may incorporate weighting. For algorithms that adaptively vary the degree of cross range averaging, we permit characterizing the variation of cross range averaging via a minimum and maximum diameter of the effective cross range

circle for the product.
MD.Qual.XrangeAvDia(1)= Minimum diameter of effective cross range averaging circle.
MD.Qual.XrangeAvDia(2)= Maximum diameter of effective cross range averaging circle.

The degree of localization of intensity or relief structure over any surface in the lidar point cloud is indicated by MD.Qual.XrangeAvDia. This can be interpreted as a cross range resolution metric since it indicates the scale of detail that can be seen in the point cloud.

Governing quality metrics
See Also

MD.Range.*

MD.Range.* parameters describe the items associated with characterizing the sensor to scene distance for the data collected to build the current data set.

20.1.110 MD.Range.Max[L1,2,3,4,5]

Brief Definition	MD.Range.Max indicates the maximum range from sensor to scene component for the current data set.
indices	none
Units/representation	Meters, float
Treatment/usage	This item is available for use when it is recognized as a useful predictor of data suitability. More explicit metrics for quality and impairment are available. Taken with MD.Range.Min, MD.Range.Max suggests the variability in the product.
Governing quality metrics	
See Also	MD.Qual.Impairments(j) MD.Qual.GSD MD.Qual.GeoLocUncH

20.1.111 MD.Range.Min[L1,2,3,4,5]

Brief Definition	MD.Range.Min indicates the minimum range from sensor to scene ground component for the current data set.
indices	None
Units/representation	Meters, Float
Treatment/usage	This item is available for use when it is recognized as a useful predictor of data suitability. More explicit metrics for quality and impairment are available. This might suggest the utility of the best quality part of the data. Taken with MD.Range.Max, MD.Range.Min suggests the variability in the

product.

Governing quality metrics

See Also

20.1.112 MD.Range.NomRgToScene [L1,2,3,4,5]

Brief Definition: MD.Range.NomRgToScene is the nominal line of sight range from the sensor to the scene for the data set.

indices: none

Units/representation: Meters, float

Treatment/usage: MD.Range.NomRgToScene should be characteristic of the dominant value of the line of sight range from the sensor to the scene in the current product. MD.Range.NomRgToScene could be the altitude of a plane carrying the sensor, distance from a fixed sensor pedestal to a scene, or the 80th percentile range value for products with significant variation in range as evidenced by the values of MD.Range.Min and MD.Range.Max. Being nominal, the range reported is characteristic of instantaneous distances between the sensor and strong returns from the intended target region. As such, it is more reasonable to report a median or similar statistic than a mean, which has the risk of being a value that is not at all associated with any part of the scene.

Governing quality metrics

See Also: MD.Range.Min MD.Range.Max

MD.Sensor.*

MD.Sensor.* parameters describe the items associated with characterizing the sensor that collected the raw data used to build the current data set.

20.1.113 MD.Sensor.ID(j) [L1,2,3,4,5]

Brief Definition: MD.Sensor.ID(j) is an identifier for the sensor(s) used to collect data in the current data set.

indices: Index, j, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of collections contributing data to the current data set. For enterprise L-1, L-2, and L-3 data sets, only one sensor may contribute to a data set. Only lidar L-4 data sets may have more than a single sensor as data sources.

Units/representation: Dimensionless, Character string length=6

Treatment/usage: MD.Sensor.ID(j) represents an identifier string that should be from an established registry for the production systems involved in generating the CMMD compliant data sets. MD.Sensor.ID uniquely identifies the series and series number of the sensor. MD.Sensor.ID(j) refers only to the sensor instrument(s) and not the platform(s). Since it is generally possible to mount

a given sensor on one or more platforms over its life, the platform for the collection is identified separately in MD.Platform.ID(j).
Note that the indexing is over collections. If the same sensor is used in more than one collection contributing to the current data set, the value of the sensor ID will be repeated for more than one value of index, j.
This is the searchable parameter that is the same as the sensor ID sub-string of MD.Meta.DS_Name.
Note for L-4 data sets: It is essential that the ordering of MD.Sensor.ID(j) in index j be the same as is used in MD.Aggregate.Parent.DS_Name(j) to consistently attribute information to each specific data set.

Governing quality metrics

See Also MD.Platform.ID(j) MD.Aggregate.Parent.SensorID(j)

20.1.114 MD.Sensor.LidarMode(j) [L1,2,3,]

Brief Definition MD.Sensor.lidarMode(j) indicates the sensor set up for collecting the data in the current data set.

indices Index, j=1,2

Units/representation Dimensionless, UInt8

Treatment/usage MD.Sensor.lidarMode(1) is a code for the class of receive demodulation scheme. The values are:

Code value	Demodulation class
0	(reserved)
1	Simple linear mode photo current detection of "waveform" return
2	Geiger mode [non-linear, saturated response to a threshold photon flux level]
3	Pulse compression heterodyne
4	Pulse compression polarization mixing
	…

MD.Sensor.lidarMode(2) is a code for the class of scan variation of the sensor optical axis direction. The values reflect geometric/temporal diversity scheme options and are:

Code value	Optical axis Scanning class
0	Direction fixed with respect to a fixed platform
1	Direction fixed with respect to platform, platform translation only.
2	Direction fixed with respected to platform, platform translation and orientation variation.
3	Direction fixed with respect to a fixed platform: Local scene

	region continuously reframed (monitoring mode)
4	Direction scanned relative to platform: minimal (single) dwells to voxels for mapping at high rate. Nadir angle varies through individual scans.
5	Direction scanned relative to platform: Fixed Conical scan to keep nadir angle constant during single scan periods and the same for consecutive scans.
6	Direction scanned relative to platform: Platform trajectory and scan articulation supports voxel revisit with forward and aft diversity (nadir angle) only to voxels.
7	Direction scanned relative to platform: Platform trajectory and Scan articulation supports voxel revisit with strong azimuth diversity as for Foliage penetration or structure/VO characterization
8	Direction scanned relative to platform: Local scene region continuously rescanned (monitoring mode), limited azimuth variation
	…

Governing quality metrics

See Also SC.Scan.Type

20.1.115 MD.Sensor.lidarType(j) [L1,2,3]

Brief Definition MD.Sensor.lidarType(j) indicates the focal plane sensor technology and modulation type of the lidar.

indices Index, j, =1,2

Units/representation Dimensionless, UInt8

Treatment/usage MD.Sensor.lidarType(j) carries a pair of codes that describes the focal plane technology and the waveform modulation used in the lidar. If different channels of the sensor use different technology, this field will refer to the channel used to produce the most detailed data in the current data set.
MD.Sensor.lidarType(1)= **Detection code**
MD.Sensor.lidarType(2)= **Modulation code**

Detection code	**Focal plane detector technology**
1	Avalanche photo diode (APD)
2	Micro-channel Plate (MCP)
3	Streak tube
4	Photo- multiplier tube
5	Charge coupled device CCD
6	CMOS array
	…

Modulation code	Modulation method
1	Pulses with no internal modulation (transmitter simply either on or off) [supports both time of flight and coherent Doppler heterodyne]
2	Pulses with Linear FM RF envelope modulation of optical carrier [pulse compression]
3	Pulses with Optical FM about center optical wavelength [pulse compression]
4	Pulses with Polarization modulation [pulse compression]
5	Pseudo-Random frequency Modulation CW
	…

Governing quality metrics

See Also MD.Aggregate.Parent.SensorType(j)

MD.Truth.*

MD.Truth.* parameters describe the items associated with characterizing any ground truthed content included in the data set.

20.1.116 MD.Truth.Dat(j) [L1,2,3,4,5]

Brief Definition MD.Truth.Dat(j) carries a list of codes defining the ground truth data incorporated in the lidar Data set

indices Index j=1,2,3,4

Units/representation Dimensionless, UInt8

Treatment/usage MD.Truth.Dat(1) carries the "Type ID", a code that tells whether the truth targets are basically point reflectors or extended features.

MD.Truth.Dat(1) =0 signifies an extended feature,
MD.Truth.Dat(1) =1 signifies point reflectors,
MD.Truth.Dat(1) =2 signifies both point reflectors and extended features used as truth in the data set

MD.Truth.Dat(2) carries the "Case ID", a code that tells whether the truth targets are native elements of the scene or were specially emplaced to serve as a calibrating reference.

MD.Truth.Dat(2) =0 signifies native scene content surveyed/ measured

to serve as truth target,
MD.Truth.Dat(2) =1 signifies the truth target (set) was emplaced into the scene as part of a ground truth/calibration activity
MD.Truth.Dat(2) =2 signifies that both the truth target (set) were emplaced into the scene as part of a ground truth/calibration activity and that native surveyed objects are rendered in the scene.

MD.Truth.Dat(3) carries the "Recognition Code", which tells how the truth target is to be recognized in a lidar point cloud.

MD.Truth.Dat(3) =0 signifies that the structural context is to be used to identify the truth target elements in the scene, [see ES.Truth.IllusKMZ.*n*(nchar) to describe how the recognition from context will be done]
MD.Truth.Dat(3) =1 signifies that the shape of the reflector/reflector array itself is the basis for recognizing the truth target returns in a lidar point cloud
MD.Truth.Dat(3) =2 signifies that the spectral signature of the truth target array is to be used as the primary means of recognizing the truth target in the lidar point cloud
MD.Truth.Dat(3) =3 signifies that the polarization signature of the truth target array is to be used as the primary means of recognizing the truth target in the lidar point cloud

MD.Truth.Dat(4) carries the "Status Code", which provides the usage status in the current data set

MD.Truth.Dat(4) =0 signifies that the truth data has not been used to calibrate the lidar geometrically or other wise
MD.Truth.Dat(4) =1 signifies that the truth data has been used to calibrate the lidar geometrically[106] (including ULEM parameters) only (no truth tags inserted, no spectral or polarization cal)
MD.Truth.Dat(4) =2 signifies that the truth data has been used to

[106] The principal geometrical update is the tile offset and Cartesian frame axis orientation. When corrections are applied to points, the axes of the Tile Cartesian frame will normally be set to East-North. When the corrections are tracked and not applied the original, likely east- north, axes defined in the early processing outputs are corrected with the extracted alignment data to their actual orientations so that the points become correct without individual point updates. If the corrections are ultimately/subsequently applied to the points, the axes would be brought to the local east- north orientations.

The phrase, " calibrate the lidar", in the context of using truth data in a lidar product, means to correct the lidar *product* by adjusting the Cartesian frame data or the points. The collection conops might involve using assessment of truth points in a cal run product to adjust the outputs of the navigational data so that the positions of truthed points after processing are closer to the true values. We would call that calibrating the *nav system*.

calibrate the lidar geometrically (including ULEM parameters) as well as to support insertion of truth tags) but no spectral, radiometric or polarization calibration has been done.
MD.Truth.Dat(4) =3 signifies that the truth data has been used to calibrate the lidar spectral response .
MD.Truth.Dat(4) =4 signifies that the truth data has been used to calibrate the lidar polarization response .
MD.Truth.Dat(4) =5 signifies that the truth data has been used to calibrate the lidar polarization and spectral response .
MD.Truth.Dat(4) =6 signifies that the truth data has been used to calibrate the lidar geometrically (including ULEM), polarization and spectral response without truth tags.
MD.Truth.Dat(4) =7 signifies that the truth data has been used to calibrate the lidar geometrically (including ULEM), polarization and spectral response with truth tags inserted.

MD.Truth.Dat(5)=0 if MD.Truth.Dat(4)=0. But when MD.Truth.Dat(4) is non-zero, the following geometry treatment methods are indicated:
MD.Truth.Dat(5)=1 if registrations were used to create a rigid volume of points that was optimally[107] registered to the truth points
MD.Truth.Dat(5)=2 if the post registration volume was treated as elastic[108] so that the truth processing could also permit a stretch/ compression of the product level cloud so that each truth point would remain anchored to the earth within the point cloud, but that all non truth points have been resampled and their positions scaled according to the truthed distances.
MD.Truth.Dat(5)=3 if truth points could be incorporated into anchoring local blocks[109] so that registrations are constrained to preserve the truth at the truth points.

Governing quality metrics

See Also ES.Point.TruthTags.*TileNum*(ipt,j) ES.Truth.IllusKMZ.*n*(nchar)

20.1.117 MD.Truth.Descr [L1,2,3,4,5]

Brief Definition MD.Truth.Descr provides a text description of the ground truthing mechanism when ground truthed returns are in the data set.

[107] Mimimized errors for all points relative to the truth locations subject to the rigid point cloud scene constraint

[108] This is intended to incorporate an assumption of continuous elasticity so that no higher scale of point aggregation than frame level point blocks may be treated as rigid.

[109] Local blocks are treated as rigid sub components of the point cloud scene

indices	none
Units/representation	Character string (20)
Treatment/usage	When a particular apparatus, such a specially crafted 3-D resolution bar array or cross beam studded with corner reflectors has been deployed to a collected scene region, that apparatus will have a name such as “ASPRS ref. Array 10 ”, or perhaps “NGA lidar bar set 4 ” If a structure native to the collected scene was surveyed, this would be called out in MD.Truth.Dat(2). In this case, a descriptive text might be put into MD.Truth.Descr such as “surveyed culvert ”, or “surveyed bldg bases ”, or “surv’d urban monmnts”, or “NGS monument AA1234 ”. In this last case, the 6 character substring AA1234 is of the form of a national geodetic survey permanent identifier. Any blank padding in text string should be trailing to facilitate meaningful alphabetical list sorting.
Governing quality metrics	
See Also	

MD.UseHist.*

MD.UseHist.* parameters describe the items associated with characterizing the use by human or software agents of the data in the current dataset. All other data in **this** dataset are unchanged by the actions documented in MD.UseHist.* (with the possible exception of MD.InfAssur.*). Populating these data items will facilitate archive management and tracking of system data usage.

20.1.118 MD.UseHist.Date(j) [L1,2,3,4,5]

Brief Definition	MD.UseHist.Date(j) provides a list of dates that the current file was accessed for use by human or software agents.
indices	Index, j, ranges from PD.LoIndx.MD_Dat to PD.HiIndx.MD_Dat, that is over number of recorded times the current dataset was accessed by human or software agents for use.
Units/representation	Days, UInt16
Treatment/usage	MD.UseHist.Date(j) is expressed using the Truncated Julian Day value. MD.UseHist.Date(j) is the number of days elapsed between midnight on May

24, 1968 and the date of documented use of the current data set.
This value is computed by NASA rules and the 16 bit value will not recycle again until the year 2147. Truncated Julian Day (TJD)[87] is tool convertible to the YYYYMMDD format for a GUI, e.g.
http://www.batse.msfc.nasa.gov/tools/date_convert/ or
http://www.usno.navy.mil/USNO/astronomical-applications/data-services/cal-to-jd-conv which supplies a Julian date (JD) that can be converted to TJD via
TJD = INT[JD − 2440000.5]
A Julian day is 24 hours or 86400 seconds exactly, ignoring any adjustment of leap seconds within the UTC calendar.
Per this CMMD, dates at the YYYYMMDD (integer day) granularity are represented as TJD in order to span a period of time of interest to the users of CMMD compliant data with a value that supports easily coded assessment of time differences and whether a given date falls within an interval of interest. It is expected that GUIs will generally convert the metadata in TJD to display as ISO 8601 compliant YYYYMMDD.

Governing quality metrics

See Also

20.1.119 MD.UseHist.Agent(j) [L1,2,3,4,5]

Brief Definition: MD.UseHist.Agent(j) provides a list of human or software agents that accessed the current file for use.

indices: Index, j, ranges from PD.LoIndx.MD_Dat to PD.HiIndx.MD_Dat, that is over number of recorded times the current dataset was accessed by human or software agents for use.

Units/representation: Dimensionless, BCS character(20)

Treatment/usage: For each access event indexed by the value of j, MD.UseHist.Agent(j) is a comma delimited list of 2 items. The first item is the name of the agent and the second indicates the location where the agent was when the data was accessed. The length of the agent substring and the location substring are not determined beyond the constraint that their total number of characters is 19. The MD.UseHist.Agent(j) entry may be post padded with blank characters as needed. These strings are informative at the human readable level, but coding with registered values registered is an optional implementation as far as this CMMD and compliance is concerned. Examples of valid strings are “John Smith,central02” and “QCscreener,QassmtCtr”.

Governing quality metrics

See Also: MD.UseHist.Date(j)

20.1.120 MD.UseHist.Purpose(j) [L1,2,3,4,5]

Brief Definition: MD.UseHist.Purpose(j) provides a list of objectives for the human or software agents that accessed the current file for use.

indices: Index, j, ranges from PD.LoIndx.MD_Dat to PD.HiIndx.MD_Dat, that is over number of recorded times the current dataset was accessed by human or software agents for use.

Units/representation: Dimensionless, BCS character(20)

Treatment/usage: For each access event indexed by the value of j, MD.UseHist.Purpose(j) is a character string that describes the usage purpose of the access event. The MD.UseHist.Purpose(j) entry may be post padded with blank characters as needed. These strings are informative at the human readable level, but coding with registered values registered is an optional implementation as far as this CMMD and compliance is concerned. Examples of valid strings are "Visual analysis " and "To generate BE DEM ". Objectives described could include the exploitation and derived product creation in the previous examples as well as quality control analysis, metadata updating, and chipping for delivering a smaller file as a subset of the data.

Governing quality metrics:

See Also: MD.UseHist.Date(j)

20.1.121 MD.UseHist.Status(j) [L1,2,3,4,5]

Brief Definition: MD.UseHist.Status(j) provides a use status condition for each access event for the current file.

indices: Index, j, ranges from PD.LoIndx.MD_Dat to PD.HiIndx.MD_Dat, that is over the number of recorded times the current dataset was accessed by human or software agents for use.

Units/representation: Dimensionless code, UInt8

Treatment/usage: For each access event indexed by the value of j, MD.UseHist.Status(j) is an integer code that describes status of the usage for the access event.

The meaning of the value codes are as follows:

MD.UseHist.Status(j) value	**Meaning**
0	immediate usage objective completed without problems
1	Use aborted and access ended (software crash)
2	Use completed but considered failure due to errors

3	Use completed with errors due to missing metadata in the dataset but outcome was acceptable
4	Use completed with only partial satisfaction due to quality of wideband data
5	Use completed but result deemed unsatisfactory due to errant data selection {should not have used this data set, but query selection failed to adequately filter this set out}

Governing quality metrics

See Also MD.ProcHist.ProdType(j)

20.2 ES.* Exploitation Support

ES.Align.*

ES.Align.* parameters describe exploitation support data related to the orientation of the local Cartesian frame in which lidar data is output with respect to the earth frame.

20.2.1 ES.Align.LocG_Vert(j_tile, j) [L2,3,4,5]

Brief Definition: ES.Align.LocG_Vert(j_tile,j) gives the local gravitational vertical at the center of the geographical region for the tile indicated by j_tile.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
The index, j, has values 1,2,3.

Units/representation: Dimensionless, Float

Treatment/usage: For each tile in a data set, indexed by j_tile, the j=1,2,3 entries of ES.Align.LocG_Vert(j_tile,j) give the ECEF x,y,z components respectively of the unit vector in the local gravitational up direction at the center latitude and longitude of the tile's horizontal coverage.
This value is not necessarily identical to the local Z direction, which is documented in ES.Align.XYZ_Flag. ES.Align.LocG_Vert is the additional information of what the geoid normal is at this location. Geoid data is from EGM2008, or the most current geoid at the time of data set generation when EGM2008 is superceded.
It is typical that building faces, "spirit" level measurements, and water flows correspond precisely to the local gravitational vertical (also known as the "plumbline") rather than the geodetic vertical direction given by the ellipsoid

	normal through a position of interest.
Governing quality metrics	Interpolation should use NGA software available at http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/egm08_wgs84.html
See Also	PD.TileMap.TileNum(j_tile) ES.Align.XYZ_Flag

20.2.2 ES.Align.X_ECEF(j_tile, j) [L2,3,4,5]

Brief Definition	ES.Align.X_ECEF(j_tile, j) gives the orientation of the tile local Cartesian frame x-axis in the ECEF frame
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles The index, j, has values 1,2,3.
Units/representation	Dimensionless, float
Treatment/usage	This lidar Conceptual Model and Metadata Dictionary defines point clouds exclusively in local Cartesian frames. In particular, the vertical direction, Z, is the same at every point in a single tile's point cloud. The local Cartesian frame may be aligned independently in each tile. Also, the X-axis direction is single valued over any single tile. ES.Align.X_ECEF(j_tile,j) is the tile-based value of the x- axis direction in ECEF coordinates. The j= 1, 2, 3 entries represent the ECEF XYZ components respectively of the local frame x-axis. The 3 numbers for any any single tile constitute a unit vector
Governing quality metrics	This is a definitional quantity not subject to error. The RSS of the j=1,2,3 values supplied must equal 1 to the precision of the floating point representation. When lidar point data is arranged within this frame, the accuracy of the alignment of any structures delineated would be described by the angle between portrayed and true directions. That is, while the tile level *Cartesian frame*, of which ES.Align.X_ECEF(j_tile, j) is a component, is subject to a combined positional and orientation error, the individual components, such as ES.Align.X_ECEF(j_tile, j) are simply errorless vector quantities. The Cartesian frame would be potentially skewed (non-orthogional) if errors could be assigned at the individual axis orientation level.
See Also	

20.2.3 ES.Align.Y_ECEF(j_tile,j) [L2,3,4,5]

Brief Definition	ES.Align.Y_ECEF(j_tile, j) gives the orientation of the tile local Cartesian frame y-axis in the ECEF frame
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles The index, j, has values 1,2,3.

Units/representation	Dimensionless, float
Treatment/usage	This lidar Conceptual Model and Metadata Dictionary defines point clouds exclusively in local Cartesian frames. In particular, the vertical direction, Z, is the same at every point in a single tile's point cloud. The local Cartesian frame may be aligned independently in each tile. Also, the Y-axis direction is single valued over any single tile. ES.Align.Y_ECEF(j_tile,j) is the tile-based value of the y- axis direction in ECEF coordinates. The j= 1, 2, 3 entries represent the ECEF XYZ components respectively of the local frame y-axis. The 3 numbers for any any single tile constitute a unit vector
Governing quality metrics	This is a definitional quantity not subject to error. The RSS of the j=1,2,3 values supplied must equal 1 to the precision of the floating point representation. When lidar point data is arranged within this frame, the accuracy of the alignment of any structures delineated would be described by the angle between portrayed and true directions. That is, while the tile level *Cartesian frame*, of which ES.Align.Y_ECEF(j_tile, j) is a component, is subject to a combined positional and orientation error, the individual components, such as ES.Align.Y_ECEF(j_tile, j) are simply errorless vector quantities. The Cartesian frame would be potentially skewed (non-orthogional) if errors could be assigned at the individual axis orientation level.
See Also	

20.2.4 ES.Align.Z_ECEF(j_tile,j) [L2,3,4,5]

Brief Definition	ES.Align.Z_ECEF(j_tile, j) gives the orientation of the tile local Cartesian frame z-axis in the ECEF frame
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles The index, j, has values 1,2,3.
Units/representation	Dimensionless, float
Treatment/usage	This lidar Conceptual Modeland Metadata Dictionary defines point clouds exclusively in local Cartesian frames. In particular, the vertical direction, Z, is the same at every point in a single tile's point cloud. The local Cartesian frame may be aligned independently in each tile. ES.Align.Z_ECEF(j_tile,j) is the ECEF tile-based value of the z- axis direction in ECEF coordinates. The j= 1, 2, 3 entries represent the ECEF XYZ components respectively of the local frame z-axis. The 3 numbers for any any single tile constitute a unit vector
Governing quality metrics	This is a definitional quantity not subject to error. The RSS of the j=1,2,3 values supplied must equal 1 to the precision of the floating point representation. When lidar point data is arranged within this frame, the

accuracy of the alignment of any structures delineated would be described by the angle between portrayed and true directions. That is, while the tile level *Cartesian frame*, of which ES.Align.Z_ECEF(j_tile, j) is a component, is subject to a combined positional and orientation error, the individual components, such as ES.Align.Z_ECEF(j_tile, j) are simply errorless vector quantities. The Cartesian frame would be potentially skewed (non-orthogional) if errors could be assigned at the individual axis orientation level.

See Also

20.2.5 ES.Align.XYZ_Flag [L2,3,4,5]

Brief Definition

ES.Align.XYZ_Flag indicates whether the earth frame local (tile level) Cartesian frame of the data is aligned with a zone based, centroid based or ECEF based east-north-up X,Y, Z axis choice.

indices

none

Units/representation

Dimensionless, UInt8

Treatment/usage

Only local Cartesian frames are used to represent point cloud data within the lidar Conceptual Model. This does not prohibit the conversion of XYZ Cartesian data to other frames such as ellipsoid referenced latitude, longitude and height above sea level or the ellipsoid when producing derived generic products or in the GUIs of exploitation tools. Part of specifying a local Cartesian frame is to indicate the orientation of its axes. When the frames are fixed to the earth, ES.Align.XYZ_Flag indicates the alignment choice that was made to arrive at the ECEF basis vector components provided in ES.Align.X_ECEF(j_tile, j), etc. The permissible values of ES.Align.XYZ_Flag encode the following options:

XYZ_Flag	Meaning
0	This is the default value. This indicates that the Z axis of the local Cartesian frame is along the mean look direction of the sensor optical axis. For any tile for which the variation of the look direction exceeds the angular field of view of that tile, this direction may differ enough from the look direction to a single lidar point within the tile that certain processing steps need to take the difference into account.
1	MGRS aligned ** X associated with zone easting direction Y associated with zone northing direction Z along ellipsoid normal at center of UTM sector containing the MGRS zone

2	Geodetic center, Ellipsoid aligned Local east- north alignment for X-Y in ellipsoid tangent plane, Z along ellipsoid normal at X=0,Y=0.
3	Geodetic center, Geoid aligned Local east- north alignment for X-Y in geoid tangent plane, Z along geoid normal at X=0,Y=0.
4	ECEF aligned axes Z is aligned with the rotational axis of the earth. X is in the equatorial plane on the prime* (0°) meridian Y is in the equatorial plane on the 90° East meridian

* For the lidar Conceptual Model, the prime meridian is the WGS-84 International Earth Rotation and Reference Systems Service used by GPS.
** MGRS northing and easting directions are those of the UTM zone they lie in.

When the point cloud is presented in a frame that is co-moving with the sensor, ES.Swath.SensorBasedCoordFlg(j) performs the function analogous to that of ES.Align.XYZ_Flag.

Governing quality metrics: Conforms to definition.

See Also: ES.Align.X_ECEF(j_tile, j) ES.Swath.SensorBasedCoordFlg(j)

ES.CollCondn.*

ES.CollCondn.* parameters describe exploitation support data visibility conditions during data collection operations for the current data set. Applies for Enterprise level 1 through 3.

20.2.6 ES.CollCondn.Obscurant [L1,2,3]

Brief Definition: ES.CollCondn.Obscurant is a flag indicating the nature of any obscurants affecting the data collection for the data set.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: ES.CollCondn.Obscurant will have the value of one of the following codes for circumstances that undermine data set utility:

Code	Meaning
0	None
1	Fog (ceiling below 100 meters)

2	High Light clouds (ceiling above 7km) (optical depth[110] less than 1)
3	Clouds (ceiling above 2 km, under 7km) (optical depth more than 3)
4	Low clouds (ceiling above 100meters, under 2km) (optical depth more than 3)
5	Light Smoke (optical depth less than 1)
6	Heavy smoke (optical depth greater than 1
7	Precipitation rain (slant path has no clouds)
8	Precipitation frozen (slant path has no clouds)
9	Terrain obscuration, natural features casting lidar shadow. This is about hills and mountains, not vegetation.
10	Feature obscuration, cultural features casting lidar shadow

For clouds, the most limiting condition should be cited if multiple layers exist.

Governing quality metrics: Fair representation. This is not quantitative data.

See Also

20.2.7 ES.CollCondn.PercentAreaAffected [L1, 2,3]

Brief Definition: ES.CollCondn.PercentAreaAffected indicates the approximate percentage of ground area in the sensor product view region voided due to obscuration

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: This is a number from 0 to 100 that indicates the degree to which the scene returns of interest have be obscured due to collection conditions. The impact of foliage on the bare earth return is ***not*** estimated in this figure.

Governing quality metrics: Fair representation. This data is not intended for quantitative exploitation, but rather for utility characterization.

See Also

ES.Offset.*

ES.Offset.* parameters describe exploitation support data that provide the offset part of a linear transformation to be performed on provided data to yield positions in meters in a Cartesian frame.

[110] Measured in mean free path units that scale to the medium.

20.2.8 ES.Offset.X(j_tile) [L2,3,4,5]

Brief Definition	ES.Offset.X(j_tile) carries the x- intercept term in a linear transformation to be applied to the lidar data in a tile to form actual position data within the Cartesian frame of the tile within a product.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
Units/representation	Meters, Float
Treatment/usage	The x-component of a 3-D Cartesian frame point, X_{meters}, is given by X_{meters}=Xdat * X_ScaleFac +X_Offset, where X_Offset is given by ES.Offset.X(j_tile) for every point in the tile indexed by j_tile. The wide band data item stored in the product that is being transformed is called "Xdat" in this definition paragraph [supplied via LD.Point.Pos.*TileNum*(ipt,1[111])] and the scaling factor is X_ScaleFac. ES.Offset.X(j_tile) should be interpreted as the offset for the first spatial coordinate of each point. This is strictly a Cartesian coordinate when angle based sensor referenced coordinates are not in effect per ES.Swath.SensorBasedCoordFlg(j) . This offset is to be distinguished from the origin of the Cartesian frame supplied for data in a tile. A point whose value of X_{meters},Y_{meters}, Z_{meters}, as defined by linear transformations of the above form, is (0,0,0), is at the origin of the Cartesian frame. That origin, in turn, has a position in the earth frame described by ES.Origin.*.
Governing quality metrics	
See Also	ES.Scale.X_Fac ES.Origin.X_ECEF(j_tile) LD.Point.Pos.*TileNum*(ipt,j) ES.WFClp.Locs.TileNum(ipi,iray,j) ES.Swath.SensorBasedCoordFlg(j)

20.2.9 ES.Offset.Y(j_tile) [L2,3,4,5]

Brief Definition	ES.Offset.Y(j_tile) carries the y- intercept term in a linear transformation to be applied to the lidar data in a tile to form actual position data within the Cartesian frame of the tile within a product.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
Units/representation	Meters, Float
Treatment/usage	The y-component of a 3-D Cartesian frame point, Y_{meters}, is given by

[111] When repeat factor not used. Shift one higher when the 1 position carries a repeat factor.

Y_{meters}=Ydat * Y_ScaleFac +Y_Offset, where Y_Offset is given by ES.Offset.Y(j_tile) for every point in the tile indexed by j_tile. The wide band data item stored in the product that is being transformed is called "Ydat" in this definition paragraph [supplied via LD.Point.Pos.*TileNum*(ipt,2^{111})] and the scaling factor is Y_ScaleFac.

ES.Offset.Y(j_tile) should be interpreted as the offset for the second spatial coordinate of each point. This is strictly a Cartesian coordinate when angle based sensor referenced coordinates are not in effect per ES.Swath.SensorBasedCoordFlg(j) .

This offset is to be distinguished from the origin of the Cartesian frame supplied for data in a tile. A point whose value of X_{meters},Y_{meters}, Z_{meters}, as defined by linear transformations of the above form, is (0,0,0), is at the origin of the Cartesian frame. That origin, in turn, has a position in the earth frame described by ES.Origin.*.

Governing quality metrics

See Also

ES.Scale.Y_Fac ES.Origin.Y_ECEF(j_tile) LD.Point.Pos.*TileNum*(ipt,j) ES.Swath.SensorBasedCoordFlg(j)

20.2.10 ES.Offset.Z(j_tile) [L2,3,4,5]

Brief Definition

ES.Offset.Z(j_tile) carries the z- intercept term in a linear transformation to be applied to the lidar data in a tile to form actual position data within the Cartesian frame of the tile within a product.

indices

The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles

Units/representation

Meters, Float

Treatment/usage

The z-component of a 3-D Cartesian frame point, Z_{meters}, is given by Z_{meters}=Zdat * Z_ScaleFac +Z_Offset, where Z_Offset is given by ES.Offset.Z(j_tile) for every point in the tile indexed by j_tile. The wide band data item stored in the product that is being transformed is Zdat [supplied via LD.Point.Pos.*TileNum*(ipt,3^{111})] and the scaling factor is Z_ScaleFac.

ES.Offset.Z(j_tile) should be interpreted as the offset for the third spatial coordinate of each point. This is strictly a Cartesian coordinate when angle based sensor referenced coordinates are not in effect per ES.Swath.SensorBasedCoordFlg(j) .

This offset is to be distinguished from the origin of the Cartesian frame supplied for data in a tile. A point whose value of X_{meters},Y_{meters}, Z_{meters}, as defined by linear transformations of the above form, is (0,0,0), is at the origin of the Cartesian frame. That origin, in turn, has a position in the earth frame described by ES.Origin.*.

Governing quality metrics

See Also: ES.Scale.Z_Fac ES.Origin.Z_ECEF(j_tile) LD.Point.Pos.*TileNum*(ipt,j) ES.Swath.SensorBasedCoordFlg(j)

ES.Origin.*

ES.Origin.* parameters describe exploitation support data that provide the origin of the local tile Cartesian frame in the global earth frame

20.2.11 ES.Origin.X_ECEF(j_tile) [L2,3,4,5]

Brief Definition: ES.Origin.X_ECEF(j_tile) is the ECEF x-coordinate of the origin of the local Cartesian frame used by the j_tileth tile.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles

Units/representation: Meters, Float

Treatment/usage: Conceptually, each tile of lidar point cloud data exists in a local Cartesian frame with a defined origin and orientation. ES.Origin.X_ECEF(j_tile), along with ES.Origin.Y_ECEF(j_tile), and ES.Origin.Z_ECEF(j_tile) determine where in the ECEF frame the origin of the local tile Cartesian frame lies.

While the full data set might have data in more than one Cartesian frame, at the tile level, only a single frame is permitted per tile. Thus, the j_tile index is supplied. If many tiles share the same ECEF x-coordinate for the origin, PD.IntTileMap.Origin.E_X.Flg can be used to avoid filling the entire j_tile long data vector with redundant information. See 10.2.1, internal tile mapping.

The following is the proceedure for obtaining ECEF values from WGS-84 geodetic coordinates, latitude, longitude and height above the ellipsoid.

```
function [X,Y,Z] = llh2xyz(lat,long, h)
  % Convert lat, long, height in WGS84 to ECEF X,Y,Z
  % lat and long given in decimal degrees.
  % altitude should be given in meters
  lat = lat/180*pi; %converting to radians
```

```
    long = long/180*pi; %converting to radians
    a = 6378137.0; % earth semimajor axis in meters
    f = 1/298.257223563; % reciprocal flattening[112]
    e2 = 2*f -f^2; % eccentricity squared

    chi = sqrt(1-e2*(sin(lat)).^2);
  X = (a./chi +h).*cos(lat).*cos(long);
  Y = (a./chi +h).*cos(lat).*sin(long);
  Z = (a*(1-e2)./chi + h).*sin(lat);
```

Governing quality metrics	This value must be correct to its precision so that error propagation has no unaccounted for bias.
See Also	ES.Offset.X(j_tile) ES.Origin.GeodtcEpoch PD.IntTileMap.Origin.E_X.Flg

20.2.12 ES.Origin.Y_ECEF(j_tile) [L2,3,4,5]

Brief Definition	ES.Origin.Y_ECEF(j_tile) is the ECEF y-coordinate of the origin of the local Cartesian frame used by the j_tileth tile.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
Units/representation	Meters, Float
Treatment/usage	Conceptually, each tile of lidar point cloud data exists in a local Cartesian frame with a defined origin and orientation. ES.Origin.Y_ECEF(j_tile), along with ES.Origin.X_ECEF(j_tile), and ES.Origin.Z_ECEF(j_tile) determine where in the ECEF frame the origin of the local tile Cartesian frame lies. While the full data set might have data in more than one Cartesian frame, at the tile level, only a single frame is permitted per tile. Thus, the j_tile index is supplied. If many tiles share the same ECEF y-coordinate for the origin, PD.IntTileMap.Origin.E_Y.Flg can be used to avoid filling the entire j_tile long data vector with redundant information. See 10.2.1, internal tile mapping. ECEF coordinates are obtained from WGS-84 geodetic coordinates by a well known transformation.
Governing quality metrics	This value must be correct to its precision so that error propagation has no unaccounted for bias.
See Also	PD.IntTileMap.Origin.E_Y.Flg

[112] If the ellipsoid is updated, new values should be used for the ellipsoid parameters. The data sets should be populated consistently with the geodetic epoch in place at the time of data set generation. See ES.Origin.GeodtcEpoch

20.2.13 ES.Origin.Z_ECEF(j_tile) [L2,3,4,5]

Brief Definition	ES.Origin.Z_ECEF(j_tile) is the ECEF z-coordinate of the origin of the local Cartesian frame used by the j_tileth tile.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
Units/representation	Meters, Float
Treatment/usage	Conceptually, each tile of lidar point cloud data exists in a local Cartesian frame with a defined origin and orientation. ES.Origin.Z_ECEF(j_tile), along with ES.Origin.X_ECEF(j_tile), and ES.Origin.Y_ECEF(j_tile) determine where in the ECEF frame the origin of the local tile Cartesian frame lies. While the full data set might have data in more than one Cartesian frame, at the tile level, only a single frame is permitted per tile. Thus the j_tile index is supplied. If many tiles share the same ECEF z-coordinate for the origin, PD.IntTileMap.Origin.E_Z.Flg can be used to avoid filling the entire j_tile long data vector with redundant information. See 10.2.1, internal tile mapping. ECEF coordinates are obtained from WGS-84 geodetic coordinates by a well known transformation.
Governing quality metrics	This value must be correct to its precision so that error propagation has no unaccounted for bias.
See Also	PD.IntTileMap.Origin.E_Z.Flg

20.2.14 ES.Origin.GeodtcEpoch(j) [L1,2,3,4,5]

Brief Definition	ES.Origin.GeodtcEpoch(j) gives the geodetic model names that determine the fine position of the Prime meridian plane that defines ECEF X=0 and the geoid datum. May impact ellipsoid parameters as well.
indices	Index, j= 1,2
Units/representation	Dimensionless, character(8)
Treatment/usage	ES.Origin.GeodtcEpoch(1) To transform CMMD compliant point data from the mandated Cartesian frame XYZ to GPS (geodetic) coordinates to the best achievable accuracy, it is necessary to know the International Reference Meridian (IRM) convention [i.e. the 0.0 degrees Longitude reference that has superceded the Greenwich observatory monument as the effective prime meridian] used in processing the CMMD compliant dataset. As the WGS-84 convention was programmed to expire in 2010, this field is necessary to distinguish data processed to the WGS-84 datum vs more current/ emerging IRM orientations. Designation options for the Geodetic epoch include “Gnnnnnnn” (which covers the G730, G873, and G1150 WGS-84 revisions, with room for an additional week count digits, or “ITRFnnnn”, which would cover ITRF2008) .

ES.Origin.GeodtcEpoch(2)
Elevations are provided in height above mean sea level as defined by the geoid. This datum is occasionally updated and the character string presented would be a a citation of "EGM96 ", "EGM08 " or successor earth gravitational models.

Governing quality metrics: ES.Origin.GeodtcEpoch(1) must be consistent with the ECEF coordinates used in the earth frame transformations built into the wide band data.

See Also

20.2.15 ES.Origin.X_MGRS(j_tile) [L2,3,4,5]

Brief Definition: ES.Origin.X_MGRS(j_tile) gives the easting value of the local Cartesian frame origin in the MGRS zone identified for the tile.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles

Units/representation: Meters, Float

Treatment/usage: This lidar conceptual model permits the data set tile-level local Cartesian frame to be chosen in terms of an MGRS zone identifier. The orientation of the ENU, or equivalently XYZ respectively, axes of the local Cartesian frame is matched to the zone for the j_tileth tile. The placement of the origin of the Local Cartesian frame may be arbitrarily chosen within the MGRS zone given by PD.TileMap.MGRS_Zone(j_tile). The horizontal position of that choice is parameterized by the easting and northing coordinates measured from the south west corner of the MGRS zone. NGA supplies a software application called GEOTRANS that will give the geodetic coordinates for the south west corner of any valid MGRS zone identifier. ES.Origin.X_MGRS(j_tile) is the easting value[113] of the local Cartesian frame origin for the j_tileth tile.

Note that both the ECEF based earth frame [e.g. ES.Origin.X_ECEF(j_tile) et al] and MGRS based earth frame [ES.Origin.X_MGRS(j_tile) et al] may be used to specify tile specific Cartesian frame origins. In the event that both sets of metadata parameters are populated in a given file, and there is inconsistency between the two coordinate sets, that is they do not refer to the same point on earth to the precision of the provided data, the user shall deem the ECEF values to be the ones intended for use. Consistent population should be enforced by validation regimes for data sets that populate both metadata item groups. That would permit the use of either set as meets the convenience of the user.

If many tiles share the same ECEF x-coordinate for the origin,

[113] This is measured from the southwest corner of the MGRS zone for the tile.

PD.IntTileMap.Origin.Z_X.Flg can be used to avoid filling the entire j_tile long data vector with redundant information. See 10.2.1, internal tile mapping.

Governing quality metrics

See Also

http://earth-info.nga.mil/GandG/geotrans/index.html
PD.TileMap.MGRS_Zone(j_tile)
PD.IntTileMap.Origin.Z_X.Flg

20.2.16 ES.Origin.Y_MGRS(j_tile) [L2,3,4,5]

Brief Definition

ES.Origin.Y_MGRS(j_tile) gives the northing value of the local Cartesian frame origin in the MGRS zone identified for the tile.

indices

The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles

Units/representation

Meters, Float

Treatment/usage

This lidar conceptual model permits the data set tile-level local Cartesian frame to be chosen in terms of an MGRS zone identifier. The orientation of the ENU, or equivalently XYZ respectively, axes of the local Cartesian frame is matched to the zone for the j_tileth tile. The placement of the origin of the Local Cartesian frame may be arbitrarily chosen within the MGRS zone given by PD.TileMap.MGRS_Zone(j_tile). The horizontal position of that choice is parameterized by the easting and northing coordinates measured from the south west corner of the MGRS zone. NGA supplies a software application called GEOTRANS that will give the geodetic coordinates for the south west corner of any valid MGRS zone identifier. ES.Origin.Y_MGRS(j_tile) is the northing value of the local Cartesian frame origin for the j_tileth tile.

Note that both the ECEF based earth frame [e.g. ES.Origin.X_ECEF(j_tile) et al] and MGRS based earth frame [ES.Origin.X_MGRS(j_tile) et al] may be used to specify tile specific Cartesian frame origins. In the event that both sets of metadata parameters are populated in a given file, and there is inconsistency between the two coordinate sets, that is they do not refer to the same point on earth to the precision of the provided data, the user shall deem the ECEF values to be the ones intended for use. Consistent population should be enforced by validation regimes for data sets that populate both metadata item groups. That would permit the use of either set as meets the convenience of the user.

If many tiles share the same ECEF y-coordinate for the origin,

PD.IntTileMap.Origin.Z_Y.Flg can be used to avoid filling the entire j_tile long data vector with redundant information. See 10.2.1, internal tile mapping.

Governing quality metrics

See Also

http://earth-info.nga.mil/GandG/geotrans/index.html
PD.TileMap.MGRS_Zone(j_tile)
PD.IntTileMap.Origin.Z_Y.Flg

20.2.17 ES.Origin.Z_MGRS(j_tile) [L2,3,4,5]

Brief Definition

ES.Origin.Z_MGRS(j_tile) gives the height value of the local Cartesian frame origin in the MGRS zone identified for the tile.

indices

The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles

Units/representation

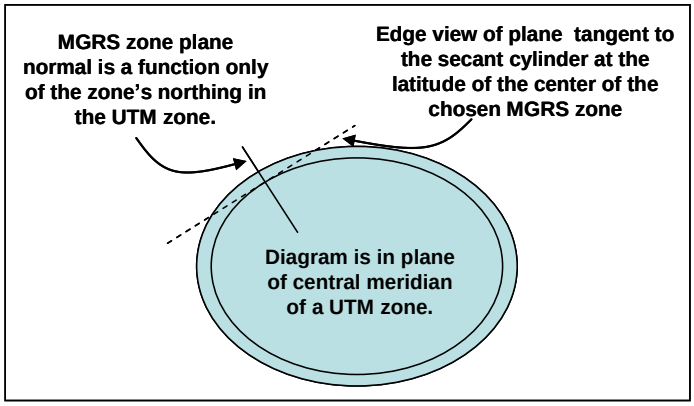
Meters, Float

Treatment/usage

The MGRS system is an extension of the UTM zone system that permits identification of smaller cells than an entire UTM zone. Strictly speaking, the UTM system uses a transverse Mercator projection of the WGS-84 ellipsoid onto a cylinder such that the center meridian scale factor is .9996. This means that the cylinder is *secant* rather than tangent to the earth. That is, the cylinder is underground at the center of a UTM zone. The cylinder can be cut along a line parallel to the transverse axis on the opposite side of the earth from a zone of interest. It can then be opened to a plane. The MGRS zones are “square” subcells on this plane. There is only one cylinder per UTM zone, so the freedom in picking the ellipsoid normal in opening the cylinder permits the plane tilt freedom only in the north-south direction.

Note that the UTM system is horizontal only. Consequently, there is no z value from UTM to be used to locate the base plane of the pt. cloud local Cartesian frame. In order to give an MGRS zone based z-value for the origin of the local Cartesian frame, we always consider that the UTM projection cylinder has been opened to a plane tangent to the secant cylinder at the normal to the ellipsoid in the central meridian plane through the center of the MGRS zone row associated with the j_tileth tile. This plane defines a z=0

plane for the MGRS zone.



ES.Origin.Z_MGRS(j_tile) is the distance in meters above this plane to the origin of the local Cartesian frame. If the local Cartesian frame origin is below this plane, ES.Origin.Z_MGRS(j_tile) will be negative.

Note that both the ECEF based earth frame [e.g. ES.Origin.X_ECEF(j_tile) et al] and MGRS based earth frame [ES.Origin.X_MGRS(j_tile) et al] may be used to specify tile specific Cartesian frame origins. In the event that both sets of metadata parameters are populated in a given file, and there is inconsistency between the two coordinate sets, that is they do not refer to the same point on earth to the precision of the provided data, the user shall deem the ECEF values to be the ones intended for use. Consistent population should be enforced by validation regimes for data sets that populate both metadata item groups. That would permit the use of either set as meets the convenience of the user.

If many tiles share the same ECEF z-coordinate for the origin, PD.IntTileMap.Origin.Z_Z.Flg can be used to avoid filling the entire j_tile long data vector with redundant information. See 10.2.1, internal tile mapping.

Governing quality metrics

See Also

PD.IntTileMap.Origin.Z_Z.Flg

20.2.18 ES.Origin.MGRS_Flag[L2,3,4,5]

Brief Definition

ES.Origin.MGRS_Flag is a code that indicates the relationship between the Local Cartesian origin choice, tiles and MGRS zones.

indices

none

Units/representation: Dimensionless, UInt8

Treatment/usage: For the purpose of this paragraph:

- A single local Cartesian frame is distinguished by a choice of origin *and* alignment of axes.
- Define a *global* Cartesian Frame to mean that each tile, representing a partition of the total data set, has coordinates defined with respect to the same single frame as all of the other tiles.
- Define *variable* Cartesian frame choices to mean that each tile has a frame that is not the same as all of the others. A tile's frame may be either unique or shared with some geographically proximate tiles.

ES.Origin.MGRS_Flag can be set to one of the exclusive choices below:

1- Global frame not attached to a large MGRS zone

If a data set with a single origin for the entire data set is composed of multiple tiles, there would be duplication of origin data values in each tile descriptor set. The offsets in each tile may or may not be different depending on how the writing application strategizes the data wordsize.

2- Global frame attached to a large enough MGRS zone to span the entire data set

3- Variable tile frames not attached to matching MGRS zones.

Would use when tiles not conveniently mosaicked to MGRS grid. Don't expect any MGRS related parameters to be populated

4- Variable Tile frames attached to matching MGRS zones

One convenient choice is to use a Cartesian Coordinate system that is horizontally positioned so that its origin is horizontally coincident with an indicated MGRS zone's South West corner. Another case of interest is to place the origin at or near the center of the horizontal space covered by an MGRS zone. When Coordinates are MGRS zone based, there may be only one MGRS zone per tile. These would be assigned via PD.TileMap.MGRS_Zone(j_tile). It is necessary to separate valid point tiles' coverage on MGRS boundaries.

Governing quality metrics: Code selected conforms to tile layout strategy

See Also: PD.TileMap.MGRS_Zone(j_tile)

ES.PDOP.*

ES.PDOP.* parameters describe exploitation support data that provide PDOP information about a data set

20.2.19 ES.PDOP.Max[L2,3,4,5]

Brief Definition	ES.PDOP.Max is the largest value of ES.PDOP.Val(j_tile) among all of the tiles in a data set.
indices	none
Units/representation	Meters, float
Treatment/usage	This is provided as a matter of convenience and is derived from the list of all PDOPS for a data set.
Governing quality metrics	
See Also	ES.PDOP.Val(j_tile)

20.2.20 ES.PDOP.Min[L2,3,4,5]

Brief Definition	ES.PDOP.Min is the smallest (best) value of ES.PDOP.Val(j_tile) among all of the tiles in a data set.
indices	none
Units/representation	Meters, float
Treatment/usage	This is provided as a matter of convenience and is derived from the list of all PDOPS for a data set.
Governing quality metrics	
See Also	ES.PDOP.Val(j_tile)

20.2.21 ES.PDOP.Val(j_tile) [L2,3,4,5]

Brief Definition	ES.PDOP.Val(j_tile) is the minimum Position Dilution of Precision for the point cloud data in a single tile.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
Units/representation	meters, Float
Treatment/usage	Position dilution of precision is a metric of the strength of geometric diversity of position measurement to reduce overall positional uncertainty. In a lidar application, some scenes may be collected from various aspects and ranges so that after registration of the separate scene point clouds under the assumption that the scene is actually identical in each collection, the position data on the collection platforms themselves can be corrected and reduced uncertainty passed on to the uncertainty of point cloud positions. Since there may be a degree of single collection diversity in the inputs to a PDOP calculation, we select the best (that is lowest), to represent a given tile. It is expected that the best PDOPs will be found for Enterprise level-4 data

sets which permit tiles to contain data from multiple flight line swaths. This is how the greatest geometric collection diversity can be created.

Populating ES.PDOP.* for levels below 4 would permit determining the improvement in PDOP in selecting various level-3 inputs a level-4 product whose purpose is to reduce geolocation uncertainty in the final point cloud.

If an enterprise level-2 or level-3 data set has had its sensor position data updated to better values and had the sensor position and/or line of sight error covariances reduced as a consequence of using information derived from a level-4 data set with good PDOP, but the points from other aspects have not been merged into the data set, we do not consider the data set be level-4. Thus PDOP is not necessarily a predictor of positional accuracy of the data.

The term is borrowed from GPS but for imagery there is a significant contribution due to the image sampling and registration which might make the purely geometric calculation less than fully descriptive. However, since the impacts of image sampling are so very variable and unpredictable as of the writing of this definition, only the geometric content is represented in this parameter. PDOP is the RMS of the x,y,z "dilutions" which are computed as terms of an error covariance matrix are but the input rows to the process are the unit vectors from a scene reference point to the sensor collection positions at the time of measuring the scene.

Governing quality metrics: Error in error estimates are not subject to consensus definition yet. Confidence in the weighting matrix used to compute PDOP would be an element.

See Also: PF.Swath.OffsetStatus

ES.Point.*

ES.Point.* parameters describe exploitation support data that provide point level information

20.2.22 ES.Point.CharCode.*TileNum*(ipt) [L2,3,4,5]

Brief Definition indices: ES.Point.CharCode.*TileNum*(ipt) carries a code characterizing a return
The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a

block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Dimensionless, UInt8

Treatment/usage: A point in a point cloud is derived from a choice of Pulse, channel (array/transmitter), element and range. There are some conditions where a valid XYZ can not be determined for a particular choice of those parameters. To permit distinguishing such in a point cloud, the metadata item ES.Point.CharCode.*TileNum*(ipt) is available.

Permitted values are:

0 –Normal valid XYZ point

1 – Geiger mode response in first possible time of pulse- element assumed to generally function properly [probable prelude to declaring a bad element]

2 – Point is probable artifact of Focal plane dysfunction (cross talk)

3 – Point is probable artifact of Pulse waveform anomaly (misplaced due to timing strategy on a non-ideal waveform)

4 – Calibration anomalous (large uncertainity in element response level)

5 – Bad element (would not populate the point normally)

6 – Geiger mode response- no response in pulse- element assumed to generally function properly (fake position generated for a graphical summary of an anomaly)

Governing quality metrics

See Also

20.2.23 ES.Point.FilterScore.*TileNum*(ipt) [L2,3]

Brief Definition: ES.Point.FilterScore.*TileNum*(ipt) gives a filter score that may be used as a thresholdable display parameter to permit an analyst to vary the number of points within a tile to display based on the probability that each displayed point is signal rather than noise or clutter.

indices: The index, “ipt”, is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation	Dimensionless, UInt8
Treatment/usage	ES.Point.FilterScore.*TileNum*(ipt) may be generated by enterprise level-2 processing applications to indicate the probability that a point represents signal of interest vice noise or clutter. For display tools that can permit varying the threshold of a parameter for which points appear on the display, this data item can give the analyst the capability to dynamically vary the filtering to suit local areas of interest at the moment. If ES.Point.FilterScore.*TileNum*(ipt) equals 000, the processing is warranting that a displayed point is without doubt a signal of interest. A safe strategy will often be that the minimum value assigned to ES.Point.FilterScore.*TileNum*(ipt) is 001. If ES.Point.FilterScore.*TileNum*(ipt) equals 255, the point is binned in the most likely to be noise or clutter category. Low values of the threshold applied to ES.Point.FilterScore.*TileNum*(ipt) should lead to a more restrictive set of displayed points and high values of the threshold should permit more points to be displayed. The display software would show all points with the value of ES.Point.FilterScore.*TileNum*(ipt) less than or equal to the threshold. If the threshold applied to ES.Point.FilterScore.*TileNum*(ipt) equals 255, all points in the tile should be visible in the display. If the threshold applied to ES.Point.FilterScore.*TileNum*(ipt) equals 000, only points with ES.Point.FilterScore.*TileNum*(ipt) equal 000 would be visible.
Governing quality metrics	
See Also	

20.2.24 ES.Point.IntensyFlg [L2,3,4,5]

Brief Definition	ES.Point.IntensyFlg indicates whether the processed intensities were in log (that is, dB) space before scaling and possible remapping to the stored values in the data set.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	ES.Point.IntensyFlg=0 or unpopulated: Values represented are proportional to photo current (linear response to inferred incident power flux) ES.Point.IntensyFlg=1: Values represented are the logarithm of scaled power flux. ES.Point.IntensyFlg applies only to the data set intensities above L-1. No value for SC.Array.CurrentRespFlg(ia) can be inferred from ES.Point.IntensyFlg.

For Geiger mode collections, the native sensor data only indicates photon reception events, not their magnitude. After some processing, the number of photon reception events for a voxel in scene space is taken as proportional to the incident power flux and this is either represented linearly (ES.Point.IntensyFlg=0) or logarithmically (ES.Point.IntensyFlg=1).

Not only does working with the logarithm of the incident intensity reduce the dynamic range, but when the uncertainty in intensity measurement is proportional to the intensity, the log makes it possible for the reported increment to be have a constant ratio with the uncertainty.

Governing quality metrics: Must fairly represent the lidar collection and processing

See Also: LD.Point.Intensity.*TileNum*(ipt) SC.Array.CurrentRespFlg(ia)

20.2.25 ES.Point.IntnsyRemap(j,k) [L2,3,4,5]

Brief Definition: ES.Point.IntnsyRemap(j,k) gives parameters for a piecewise linear remapping that has been applied to reduce the wordsize needed to convey the intensity data.

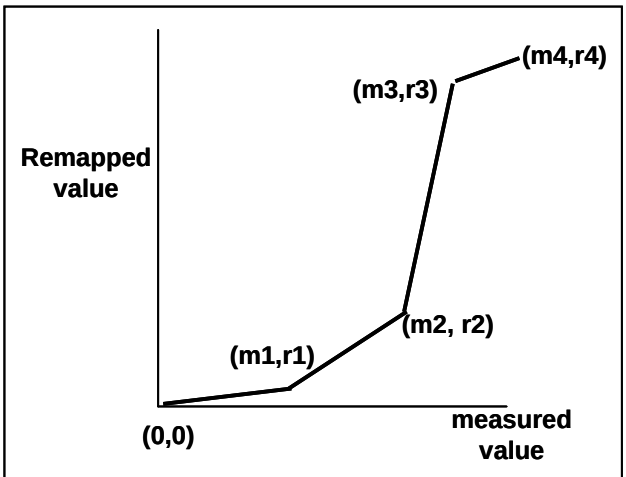
indices: Index, j, ranges from PD.LoIndex.Remap to PD.HiIndx.Remap, that is, over the number of linear segments in the remapping function as described in this paragraph
k=1 for measured intensity values, 2 for remapped intensity value

Units/representation: Dimensionless, UInt16

Treatment/usage: When ES.Point.IntnsyRemap(j,k) is not populated, the intensity data in LD.Point.Intensity.*TileNum*(ipt) are to be interpreted as un-remapped and subject only to the impact of the value of ES.Point.IntensyFlg when it is populated.

The native dynamic range of the intensity might exceed that which can be carried in an 8 bit word. The piecewise linear remapping provides a good way to pack the original intensity data into a lidar data set. Essentially, several linear segments that are continuous make up the complete remap function. Each segment is determined by its terminal point and is assumed to start at the terminal point of the preceding segment. The assumption is that (0,0) is always the first point of the first segment of the remap transformation, but it is not supplied. In the diagram below, the horizontal axis spans the range of the scaled measurements. By “scaled”, we mean that there is a single constant that converts all of the scaled values to some absolute intensity value in physical units. But limitations in lidar design and operational issues make the likelihood of encountering radiometrically calibrated data very low. That

is, the conversion constant is neither generally known nor needed. Nevertheless, consistently scaled values, expressible as positive integers to meaningful precision[114] are practical. This is what we refer to as the "measured value" of intensity.



Each segment of the piecewise linear remap function is determined by the pair of integers corresponding to the measured relative intensity (globally linear function) and the remapped intensity mi and ri respectively.

Thus, ES.Point.IntnsyRemap(1,1) is the measured intensity value of the terminal point of the first segment and ES.Point.IntnsyRemap(1,2) is the remapped intensity value (that is, after compression by the remap function) of the terminal value of the first segment of the over all remap function. These correspond to (m1,r1) of the diagram respectively. Similarly, ES.Point.IntnsyRemap(2,1) and ES.Point.IntnsyRemap(2,2) correspond to (m2,r2).

Note that consistency requires:

- The maximum value of ES.Point.IntnsyRemap(j,2) not exceed the capacity of the wordsize of LD.Point.Intensity.*TileNum*(ipt). Commonly this would be 8 bits.

[114] This means that a unit step is of the order, or greater than, the uncertainty in the underlying scaled intensity measurement.. The granularity of steps should be determined by making the full dynamic range contain all of the measured values that are resolved by the measuring method in steps that equal or exceed the uncertainty.

- The number of measured intensity steps in the interval from 0 to the maximum value not exceed[115] 65536 (48 dB of dynamic range).

The same piecewise linear remap function is used for all tiles in a single data set.
The remap function is a representation of a lookup table whereby the stored value of intensity at a point can be converted to a value that has the same conversion factor to the physical intensity as any other point. The stored values may have been reduced to a smaller word size by relying on this remap function. Using the piecewise remap, however, could mean that the stored numbers have a different linear transformation to physical values proportional to intensity depending on which segment of the remap the stored value lies. See also, the discussion of Intensity in LD.Point.Intensity.*TileNum*(ipt).

Governing quality metrics: Correct values must be populated.

See Also: LD.Point.Intensity.*TileNum*(ipt)

20.2.26 ES.Point.LookSpread.*TileNum* [L2,3,4,5]

Brief Definition: ES.Point.LookSpread.*TileNum*(ipt) is the angle between the look directions between the points in a tile that have the greatest difference in line of sight.

indices: none
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: radians, single float

Treatment/usage: The points in a single tile may have been collected from a range of viewing directions by the sensor. To characterize the variation in the line of sight from sensor to points within a tile, we introduce ES.Point.LookSpread.*TileNum*(ipt). It is the angle between the lines of sight for the points with the most divergent lines of sight in tile, *TileNum*. This is a convenient adjunct to ES.Align.XYZ_Flag=0, which indicates that the Z axis of the local Cartesian frame is along the mean line of sight for the points in a tile.

[115] If and when a sensor is capable of exceeding this performance, we would add a wordsize parameter, PD.Wordsize.Remap, that would have a default value of 16 bits. That would support backwards compatiblility to this version of the CMDD.

Governing quality metrics

See Also ES.Align.XYZ_Flag

20.2.27 ES.Point.LM_SigmaR.*TileNum*(ipt) [L2,3,4,5]

Brief Definition ES.Point.LM_SigmaR.*TileNum*(ipt) is the linear mode range uncertainty on a point by point basis.

indices The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation Meters, float

Treatment/usage In linear mode, the range from the sensor to a point depends on the round trip time assigned to the point and the time to range conversion process. ES.Point.LM_SigmaR.*TileNum*(ipt) combines the effect of the time uncertainty and the time to range conversion to create a net range uncertainty on a pulse-ray, that is a point, basis. The main source of time uncertainty is the shape of a waveform peak, with a secondary contributor of the fast time recording time step. The conversion to range includes index of the refraction impacts of uncertain propagation speed and possible ray bending.

Governing quality metrics

See Also ES.Point.LM_SigmaT(ipt)

20.2.28 ES.Point.LM_SigmaT.*TileNum*(ipt) [L2,3,4,5]

Brief Definition ES.Point.LM_SigmaT.*TileNum*(ipt) is the linear mode round trip time uncertainty on a point by point basis.

indices The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation	Seconds, float
Treatment/usage	ES.Point.LM_SigmaT.*TileNum*(ipt) is provided as an adjunct to ES.Point.LM_SigmaR.*TileNum*(ipt) to permit the backing out of the time to range conversion uncertainty on a point basis. Both items tend to vary from point to point, but many returns in the scene may share similar return waveform envelopes while a different set of returns could share similar time to range conversion uncertainty. To assure the ability to analyze any particular point, the data fields ES.Point.LM_SigmaT.*TileNum*(ipt) and ES.Point.LM_SigmaR.*TileNum*(ipt) are both defined. The population plan will determine whether the capabilities afforded by having both data items will be supported in any particular data set.
Governing quality metrics	
See Also	ES.Point.LM_SigmaR(ipt) ES.Swath.GM_SigmaT

20.2.29 ES.Point.PulseNum.*TileNum*(ipt, j) [L2,3,4,5]

Brief Definition	ES.Point.PulseNum.*TileNum*(ipt, j) provides the pulse number (also referred to as the pulse index) for which a specific point was generated.
indices	The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile). See detailed discussion of "ipt" in PF.Point.IrayNum.*TileNum*(ipt). *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.PulseNum.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.PulseNum.*TileNum*.Flg >0, i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume.
Units/representation	Dimensionless, Variable length Integer byte length given by PD.Wordsize.Pulse Precision returned by DAL in DAL implementations of the lidar data model. Choose the lowest workable value for a given data set to reduce data volume.
Treatment/usage	ES.Point.PulseNum.*TileNum*(ipt, j) provides the value of the IPP count (aka pulse index) for which a point's raw data was recorded. Lidar points naturally result from a return for a specific pulse number and "ray" [there is a unique ray for each detector element]. Thus lidar points, because, they

originate from a specific pulse, could have the pulse number to which they are due be a populated data item. More subtlely, a synthetic lidar point might also have a specific pulse number assigned to it, despite the fact that synthetic points are generally synthesized from multiple points, some of which may have had different source pulses. [116] Regardless of whether a point is synthetic or not, the processor documents the pulse for it when it has (or has been given) a uniquely defined value by populating the value of ES.Point.PulseNum.*TileNum*(ipt, j).

To support a reduction in byte volume, the IPP counting for each point tile may be started at 0. {Actually, each of potentially several intra-tile time segments can be started at 0, permitting greater reduction in the IPP index values that need to be populated.} To obtain the correct global IPP value (counted from the start of the collection) for a point, one would need to add the correct IPP index offset from PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k). The procedure to do this is as follows:

Since a point has been selected, the point oriented *TileNum* and tile specific value of ipt are known.

1. Access the PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) array with the matching value of *TileNum*.
2. In level-2 data sets, the range of j_parent is one value, so we do not have to worry about staying on the correct "parent" row of the array. We need to first figure out which time segment the point in row "ipt" belongs to. This is done by stepping through the values of the j index and checking for values of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,1) and PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,2) that bracket or equal "ipt". The value of j so determined is the time segment index of interest. We call it "j".
3. Every point in ES.Point.PulseNum.*TileNum*(ipt, j) has been populated with the number of IPP counts past PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,3) corresponding to its collection IPP. So the global pulse number for the point in *TileNum* with (local) point index"ipt" is given by

[116] It is recommended that time tagging a synthetic lidar point be done by associating the time of the contributing real lidar point which is closest to the position of the synthetic point. That position will generally be quite close to the centroid of the contributors to the synthetic point. This strategy is useful in supporting meaningful line of sight association with synthetic points, where there is no actual line of sight defined. This is particularly useful where multi-dwells are used and the mean time of collection may be disconnected from all of the contributing points.

ES.Point.PulseNum.*TileNum*("ipt", *j*[117]) +
PD.TileMap.StartPulseNum.*TileNum*(j_parent, "j",3).

So if the offset wordsize control strategy is used, PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) must be populated. If it is not, global values of the IPP count for a point are to be assumed for ES.Point.PulseNum.*TileNum*(ipt, j).

Governing quality metrics: Value must be the correct assignment of pulse number to each point in the data.

See Also: PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) PF.Point.IrayNum.*TileNum*(ipt) PD.IntTileMap.PulseNum.*TileNum*.Flg PD.Wordsize.Pulse

20.2.30 ES.Point.RtnsCode.*TileNum*(ipt, j) [L2,3,4,5]

Brief Definition: ES.Point.RtnsCode.*TileNum*(ipt,j) encodes information about the return that results in lidar point (ipt)

indices: The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).
The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.RtnsCode.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.RtnsCode.*TileNum*.Flg >0, i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Dimensionless UInt8

Treatment/usage: Lidar returns may result from Geiger mode time of incidence reports, decidedly point-like linear mode wave-form peaks, or from more spread out linear mode returns. Depending on the quench timing in Geiger mode and the transmitted power and receive window in linear mode, multiple returns for a single pulse and channel may be recorded in the lidar sensor. Thus, there is

[117] We use italicized "*j*" to both distinguish this from the "j" value for the time segment just determined and to indicate the need to resolve the actual value if repetition factors are in use for ES.Point.PulseNum.*TileNum*(ipt, j).

information to be relayed regarding which of the possible returns for a pulse-channel a particular point represents. Also, the total number of returns reported for the same channel-pulse-ray, and how many of the preceding returns for a given channel-pulse-ray were point-like can be of value in interpreting the lidar measurements of a scene. The following table shows how the return number, number of lower numbered returns that are point-like and the total returns for the same pulse-ray as a given point can be encoded into a single 8 bit word for up to 7 returns. This is an embedded Excel spread sheet that can be opened by double clicking and then scrolled down to the last code, 196.

Code	return#	lower returns that are points	Total returns
0	0	0	0
1	1	0	1
2	1	0	2

For a lidar that is operated in Geiger mode with a single quench per pulse, there is no information to be carried by ES.Point.RtnsCode.*TileNum*(ipt) because only a single return per pulse-ray is possible. In such a case, ES.Point.RtnsCode.*TileNum*(ipt) may be omitted without loss of capability. Since data for a single pulse but different channels will be reported on different “rows” (value of ipt) for tiles with mixed channel reporting, the return number encoded in ES.Point.RtnsCode.*TileNum*(ipt) must be for the channel corresponding to the value of ipt. For dedicated channel tiles, all rows have the same channel and the channel associated with ES.Point.RtnsCode.*TileNum*(ipt) is clearly derived from the tile channel value.

Governing quality metrics

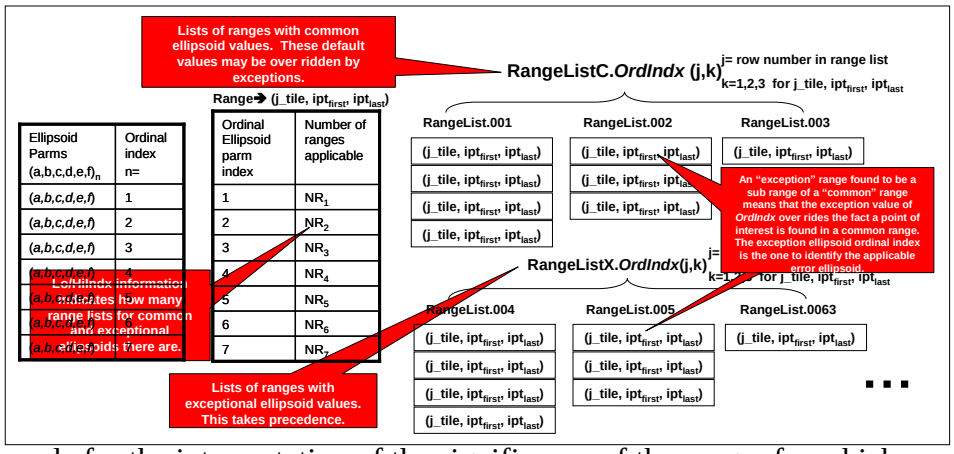
See Also

ES.Point.CharCode.*TileNum*(ipt) ES.WFClp.RtnsCode.*TileNum*(ip,iray)
PD.IntTileMap.RtnsCode.*TileNum*.Flg

20.2.31 ES.Point.SigmaXYZ(j,k) [L2,3,4,5]

Brief Definition	ES.Point.SigmaXYZ(j,k) is a list of pre-computed point error ellipsoid parameter value sets that apply to specific ranges of points in the data set.
indices	Index, j ranges from PD.LoIndx.Point.SigmaDat to PD.HiIndx.Point.SigmaDat, that is, from 1 to the number of pre-computed error ellipsoids. Index, k=1,2,3,4,5,6 for sigmaX, SigmaY,SigmaZ, rhoxy, rhoxz, rhoyz , respectively
Units/representation	Meters, Float
Treatment/usage	While the capability to do tool based error propagation on specific manually selected points from a lidar point cloud data set is supported by the metadata model, it may enhance utility when production latency is not an issue to pre-compute and store error ellipsoids so that this data can be made readily available via a look up rather than requiring an analyst to exercise a tool to obtain point geolocation uncertainties. To support this latter functionality, ES.Point.SigmaXYZ(j,k) may be populated. When this data item is populated, each row (determined by the value of j) represents the error ellipsoid of some number of points, which could range from one to very many. Thinking of the index, j, as an ordinal index indicating which ellipsoid from a list of pre-computed ellipsoids for a data set is being referenced, we map the error ellipsoid to a range of points within the data set via PD.Point.RangeListC.*OrdIndx*(j,k) and PD.Point.RangeListX.*OrdIndx*(j,k). Let us introduce the term “common” error ellipsoid to represent an error ellipsoid shared by points in a broad range of points in a data set. Such “common” error ellipsoids should be placed at lower values of j in ES.Point.SigmaXYZ(j,k). Let us also introduce the term “exceptional” error ellipsoid to represent values of the error ellipsoid that appear over short ranges of point index that are embedded within a range with a different, “common”, value of the error ellipsoid. More than one range of points may share the same error ellipsoid. It is also recommended that the error ellipsoids be treated somewhat as discretes so that a few error ellipsoids may be used rather than a large near continuum of error ellipsoids. Being error data, tremendous precision in the elements of the error ellipsoid is of little value. Essentially, only the last significant digit of a position should be spanned by error data. The substring, *OrdIndx* is an evaluated piece of the tag name that is a character representation of a value of the j index of ES.Point.SigmaXYZ(j,k). So, for example, the second error ellipsoid in the ES.Point.SigmaXYZ(j,k) list (where j=2) has a corresponding list of point ranges named, for example, PD.Point.RangeListC.002(j,k). On the other hand, the j=10 set of ellipsoid

parameters may have been computed for a sub-range of the range carried by PD.Point.RangeListC.002(2,k). In that case, the j=10 ellipsoid would be an exception to the more common ellipsoid value generally found in the range PD.Point.RangeList**C**.002(2,k). It would apply to point index ranges provided by PD.Point.RangeList**X**.010(j,k). If a point falls in a range for some PD.Point.RangeListX.*OrdIndx*(j,k) then the indicated exceptional value (the exception error ellipsoid data is indicated by the value of *OrdIndx)* over rides the fact that the point also falls in a range for PD.Point.RangeListC.*OrdIndx*(j,k). In other words, the broad ranges for common values of SigmaXYZ would be permitted to cover a number of smaller ranges that need to be separately identified as exceptions. One would check a point of interest for presence on exception lists first. A point's absence on any PD.Point.RangeListX.*OrdIndx*(j,k) "exception" list would mean it shares the SigmaXYZ with the other points in the broad range that includes it in PD.Point.RangeListC.*OrdIndx*(j,k). This design permits the "data set writer" application to choose the exception ranges and use a data sort that supports easily enumerating them. This minimizes the data volume to map error ellipsoids to all the points in a data set. The diagram below lays out the logical relationships between the various pre-computed error ellipses for points in the earth frame, the point ranges to which they apply and the indexing and naming of the metadata items carrying this information.



A code for the interpretation of the significance of the ranges for which common values of the ellipsoid are presented appears in PF.CommonSigma.AggLevel.

Governing quality metrics

There is no error of the error metric. However, the error ellipsoids should be of a size commensurate with the precision[118] that points are populated with. Furthermore, excessive point density in a data set leads to the listing of points whose positions cannot be assured to be distinct from the positions of their neighbors with high probablility. In other words, if the error ellipsoids of many points overlap that of their neighbors, reporting their positions is gratuitous. Intensity reporting strategies may permit reducing data set size so that when the point earth frame error ellipsoids are known, the point lists are not needlessly dense.

See Also

PF.CommonSigma.AggLevel PD.Point.RangeListC.*OrdIndx*(j,k) PD.Point.RangeListX.*OrdIndx*(j,k) PD.HiIndx Point.SigmaDat

20.2.32 ES.Point.TruthTags.*Tilenum*(ipt,j) [L2,3,4,5]

Brief Definition

ES.Point.TruthTags.*TileNum*(ipt,j) enables points to be labled as associated with known truthed locations vice merely lidar sensor system derived locations.

indices

The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

The index, j, has a default range value of 1:2 (or 0:1 as implementers prefer) for the case of PD.IntTileMap.TruthTags.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.TruthTags.*TileNum*.Flg >0, there are three values in the range of j, e.g, 1:3. This index supports repetition factor use to reduce data volume.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a "tile". This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation

Treatment/usage

Per section 10.2.1.5, a tiled item that has long repeated sequences of the same value may be stored as a repeat count and a value. Since only a small number of the points in a lidar data set can be truthed points, a vector of values

[118] This is a reco to keep the wordsize for XYZ data down to avoid carrying meaningless precision. Without knowledge of values of ES.Point.SigmaXYZ(j,k), this can not be done with justification. Once this has been computed, it may as well be reported.

indicating truthed position vice derived position would be dominated by long strings of indicator set to the code for derived.
Thus, in practice, the PD.IntTileMap.TruthTags.*TileNum*.Flg will always be populated to indicate the "compressed" array ES.Point.TruthTags.*TileNum*(ipt,j) actual row count. The code for truth is set as follows:
ES.Point.TruthTags.*TileNum*(ipt,2)=0 means that the point position is not associated with a ground truth location.
ES.Point.TruthTags.*TileNum*(ipt,2)=1 means that the point position is associated with a ground truthed location and the association of the point and truth location was highly certain.
ES.Point.TruthTags.*TileNum*(ipt,2)=2 means that the point position is associated with a ground truthed location and the association of the point and truth location was moderately certain.
ES.Point.TruthTags.*TileNum*(ipt,2)=3 means that the point position is associated with a ground truthed location and the association of the point and truth location was weakly certain. This would be the case when a high density of points lies in the vicinity and any of the nearby points in a local cluster could likely have been the truth point. In such a case, the impact of mistaking the ground truthed point results in only minor error due to the effective equivalent locations of the confusing points.
ES.Point.TruthTags.*TileNum*(ipt,3) will be set to the "itp" value of the truthed data, thus indicating which truth data values are associated with the points tagged as corresponding to truthed locations. Note that LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) always has the lidar based point position, even when the truthed point flag is indicated. The truthed geodetic position for such a flagged lidar point is in ES.Truth.Pos(itp,j).

Until the state of the art achieves automated truth point identification, we can expect that the addition of ES.Point.TruthTags.*TileNum*(ipt,j) to any lidar data set will be based on manual assessment of enterprise level-2 data[119], and may be executed via editing rather than processing. Furthermore, ES.Point.TruthTags.*TileNum*(ipt,j) should only be populated when ES.Truth.Pos(itp,j) has been populated, as identifying returns as truth is only feasible when the truth is available.

Once truth tags have been populated, finding truthed data within a tile is straightforward. If there are many tiles, and only a few have points that have

[119] Manual assessment is normally an indicator that a data item, and thus the data set that contains it, is enterprise L-5. However, we make an exception for identification of truth data points on the grounds that it is part of the production process and is not interpreting points of unknown origin.

	been associated with truthed scene locations, one could identify the tiles with truth by searching for children[120] of keywords ES.Point.TruthTags and noting the values of *TileNum* that are found by the DAL.
Governing quality metrics	
See Also	PD.IntTileMap.TruthTags.*TileNum*.Flg ES.Truth.Pos(itp,j)

ES.SAL.*

ES.SAL.* parameters describe synthetic aperture output data, when applicable.

20.2.33 ES.SAL.GridAxes(j) [L2]

Brief Definition	ES.SAL.GridAxes(j) gives the directions of the row (i) -axis direction and cross-row (j)- axis direction in the ECEF frame of the output data grid, LD.SAL.DataGrid.*ChNum.TileNum*(i,j,k). Applies for Synthetic Aperture Lidar processing only.
indices	j=1,2,3 ECEF row direction x,y,z j=4,5,6 ECEF cross row direction x,y,z (normal wil be cross product)
Units/representation	
Treatment/usage	*Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	

20.2.34 ES.SAL.GridType [L2]

Brief Definition	ES.SAL.GridType indicates range-Doppler vs along track-cross track output coordinates
indices	j=1,2,3 ECEF row direction x,y,z j=4,5,6 ECEF cross row direction x,y,z (normal wil be cross product)
Units/representation	
Treatment/usage	=1 for range, Doppler (no keystone correction), =2 for along, cross track(output coordinates are Cartesian) output grid *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

[120] These are ES.Point.truthTags.*TileNum* .

20.2.35 ES.SAL.IPR(j) [L2]

Brief Definition	ES.SAL.IPR(j) Indicates the output plane IPR -3dB width
indices	j=1 for range like IPR -3dB width, j=2 for cross range like
Units/representation	
Treatment/usage	=1 for range, Doppler (no keystone correcton), =2 for along, cross track(output coordinates are Cartesian) output grid *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.2.36 ES.SAL.ORP(j) [L2]

Brief Definition	ES.SAL.ORP(j) provides the postionof the output reference point in the complex SAL data grid
indices	Index, j=1 to 5 (1,2) row, column of ORP in DataGrid (3,4,5) ECEF x,y,z of output reference point.
Units/representation	Dimensionless (i.e. samples) Float
Treatment/usage	"upper left" pixel of 4th quadrant data grid is at (0.0, 0.0) *Stub, Details pending, Need to support tree and table data in 9.2 and PD.Lo/hiIndx*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.2.37 ES.SAL.SL_RgAxis(j) [L2]

Brief Definition	ES.SAL.SL_RgAxis(j) indicates the output plane range sidelobe axis direction.
indices	Index, j=1,2
Units/representation	Dimensionless, float
Treatment/usage	Index, j=1component in range like (i) direction , i=2 component in cross range like (j) direction of output plane *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.2.38 ES.SAL.SL_AzAxis(j) [L2]

Brief Definition	ES.SAL.SL_AzAxis(j) indicates the output plane range sidelobe axis direction.
indices	Index, j=1,2
Units/representation	Dimensionless, float

Treatment/usage	Index, j=1component in range like (i) direction , i=2 component in cross range like (j) direction of output plane *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.2.39 ES.SAL.StepSize(j) [L2]

Brief Definition	ES.SAL.StepSize(j) provides the range like and cross range like sample step size in the SAL output data grid
indices	Index, j=1,2
Units/representation	Meters, Float
Treatment/usage	*Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	

ES.Scale.*

ES.Scale.* parameters describe the scaling part of the linear transformation that converts stored lidar point position data to meters.

20.2.40 ES.Scale.X_Fac [L2,3,4,5]

Brief Definition	ES.Scale.X_Fac is the coefficient of the first coordinate (x in the Cartesian case) axis data in the linear transformations that produce coordinate values in meters (or angle units) from stored first coordinate components of lidar point position.
indices	none
Units/representation	As implied by ES.Swath.SensorBasedCoordFlg(j), float
Treatment/usage	In order to allow minimization of the data volume, lidar point position data is stored such that it is restored to SI unit positions (or cosines if indicated by ES.Swath.SensorBasedCoordFlg(j)) referenced to the local frame origin via a linear transformation of the form: Xpos=Xdat * X_Fac +X_Offset In the above transformation, Xdat is supplied via LD.Point.Pos.*TileNum*(ipt,1). Depending on ES.Swath.SensorBasedCoordFlg(j), this could be a Cartesian x-coordinate, or an angle measure. While we permit the X_Offset to vary from tile to tile in order to support use of smaller wordsizes for LD.Point.Pos.*TileNum*(ipt,1), the first coordinate scaling is global for the data

set. This makes the relative distances in the pre-scaled data uniform and permits trivial metadata support for unit scaling on a global basis. If ES.Scale.X_Fac is not populated, it is assumed to be 1.0E_00 meters/step.

Anywhere in the above description where the use of repeat (internal tile mapping) flags causes the j index for the first coordinate to shift, the appropriate accomodation should be made. By the same token, in a case that the coordinate system, per ES.Swath.SensorBasedCoordFlg(j), is not Cartesian, the correct first coordinate name should be understood to replace "X".

Governing quality metrics: The value must be the exact value used by the processor to generate LD.Point.Pos.*TileNum*(ipt,1)

See Also: LD.Point.Pos.*TileNum*(ipt,j) ES.Offset.X(j_tile) ES.Swath.SensorBasedCoordFlg(j)

20.2.41 ES.Scale.Y_Fac[L2,3,4,5]

Brief Definition: ES.Scale.Y_Fac is the coefficient of the second coordinate (y in the Cartesian case) axis data in the linear transformations that produce coordinate values in meters (or angle units) from stored second coordinate components of lidar point position.

indices: none

Units/representation: As implied by ES.Swath.SensorBasedCoordFlg(j), float

Treatment/usage: In order to allow minimization of the data volume, lidar point position data is stored such that it is restored to SI unit positions (or cosines if indicated by ES.Swath.SensorBasedCoordFlg(j)) referenced to the local frame origin via a linear transformation of the form:

Ypos=Ydat * Y_Fac +Y_Offset

In the above transformation, Ydat is supplied via LD.Point.Pos.*TileNum*(ipt,2). Depending on ES.Swath.SensorBasedCoordFlg(j), this could be a Cartesian y-coordinate, or an angle measure. While we permit the Y_Offset to vary from tile to tile in order to support use of smaller wordsizes for LD.Point.Pos.*TileNum*(ipt,2), the second coordinate scaling is global for the data set. This makes the relative distances in the pre-scaled data uniform and permits trivial metadata support for unit scaling on a global basis. If ES.Scale.Y_Fac is not populated, it is assumed to be 1.0E_00 meters/step.

Anywhere in the above description where the use of repeat (internal tile mapping) flags causes the j index for the first coordinate to shift, the appropriate accomodation should be made. By the same token, in a case that

	the coordinate system, per ES.Swath.SensorBasedCoordFlg(j), is not Cartesian, the correct second coordinate name should be understood to replace "Y".
Governing quality metrics	The value must be the exact value used by the processor to generate LD.Point.Pos.*TileNum*(ipt,2)
See Also	LD.Point.Pos.*TileNum*(ipt,j) ES.Offset.Y(j_tile)

20.2.42 ES.Scale.Z_Fac [L2,3,4,5]

Brief Definition	ES.Scale.Z_Fac is the coefficient of the third coordinate (z in the Cartesian case) axis data in the linear transformations that produce coordinate values in meters (or angle units) from stored second coordinate components of lidar point position.
indices	none
Units/representation	As implied by ES.Swath.SensorBasedCoordFlg(j), float
Treatment/usage	In order to allow minimization of the data volume, lidar point position data is stored such that it is restored to SI unit positions (or cosines if indicated by ES.Swath.SensorBasedCoordFlg(j)) referenced to the local frame origin via a linear transformation of the form: Zpos=Zdat * Z_Fac +Z_Offset In the above transformation, Zdat is supplied via LD.Point.Pos.*TileNum*(ipt,3). Depending on ES.Swath.SensorBasedCoordFlg(j), this could be a Cartesian z-coordinate, or an angle measure. While we permit the Z_Offset to vary from tile to tile in order to support use of smaller wordsizes for LD.Point.Pos.*TileNum*(ipt,3), the third coordinate scaling is global for the data set. This makes the relative distances in the pre-scaled data uniform and permits trivial metadata support for unit scaling on a global basis. If ES.Scale.Z_Fac is not populated, it is assumed to be 1.0E_00 meters/step. Anywhere in the above description where the use of repeat (internal tile mapping) flags causes the j index for the first coordinate to shift, the appropriate accomodation should be made. By the same token, in a case that the coordinate system, per ES.Swath.SensorBasedCoordFlg(j), is not Cartesian, the correct third coordinate name should be understood to replace "Z".
Governing quality metrics	The value must be the exact value used by the processor to generate LD.Point.Pos.*TileNum*(ipt,3)
See Also	LD.Point.Pos.*TileNum*(ipt,j) ES.Offset.Z(j_tile)

ES.Scan.*

ES.Scan.* parameters describe the scan level and pre-adaptive processing features of exploitation support data.

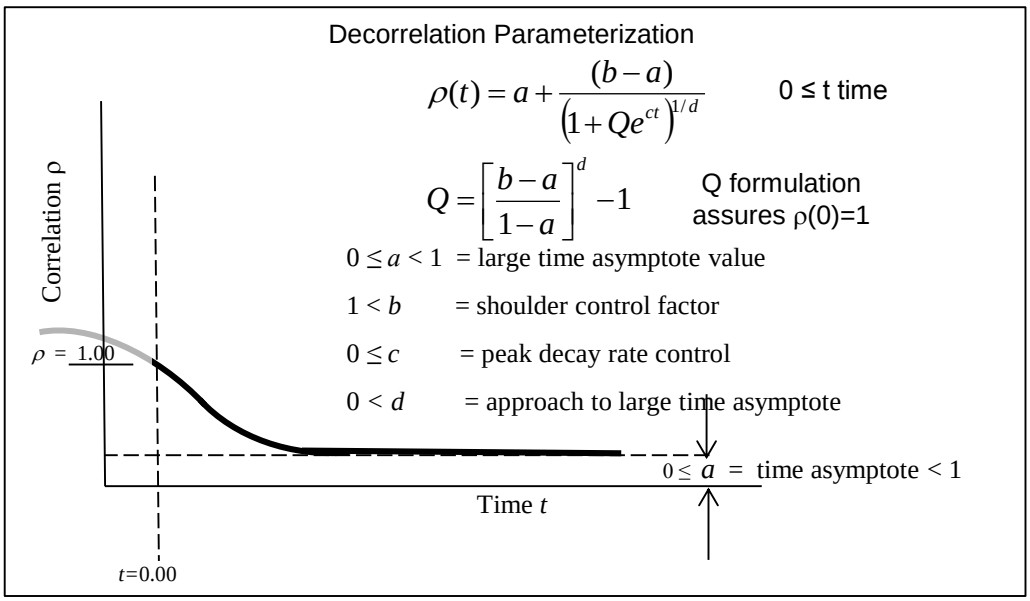
20.2.43 ES.Scan.CorrParms.Alpha(j,k) [L1,2,3,4,5]

Brief Definition	ES.Scan.CorrParms.Alpha(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal x-reference direction at the sensor.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty. Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.
Units/representation	Variable, single float
Treatment/usage	The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time. ES.Scan.CorrParms.Alpha(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor x-axis based direction cosine as determined at each moment of sampling the error covariance matrix. A more complete description of the look axis geometric parameters is available in ES.Scan.SigmaAlpha(j), SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), and in 10.1.2.3. The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals. It may be that the attitude control system filter uses a constant integration time and that all of the samples of ES.Scan.CorrParms.Alpha are the same. [In this case there will be only one set of ES.Scan.CorrParms.Alpha correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the attitude control uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. The quantities given by ES.Scan.CorrParms.Alpha describe the error

correlation of a net LOS measure that includes the effect of the orientation of the sensor itself and the steering system that directs the beam optical axis relative to the sensor. While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak error) calculation of the correlations. The auto correlation of cosine Alpha is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.Alpha(j,1) = a, the large time asymptotic value

ES.Scan.CorrParms.Alpha(j,2) = b, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.Alpha(j,3) = c, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.Alpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.Alpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \geq 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.Alpha(j,k) where index, j, is index of the first snapshot in a consecutive pair of snapshots.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j) ES.Swath.CorrParms.Alpha(j) SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) ES.Scan.CorrParmsT.Alpha(j,k)

20.2.44 ES.Scan.CorrParms.Beta(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParms.Beta(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal y-reference direction at the sensor.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation

Variable, single float

Treatment/usage

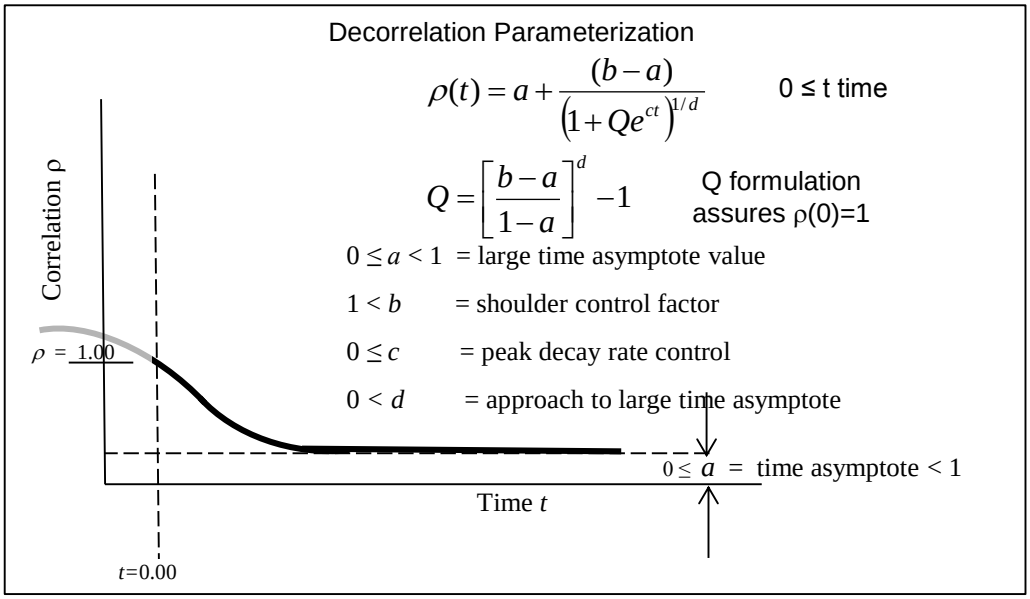
The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative

distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.Beta(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor y-axis based direction cosine as determined at each moment of sampling the error covariance matrix. A more complete description of the look axis geometric parameters is available in ES.Scan.SigmaBeta(j), SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), and in 10.1.2.3. The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals.
It may be that the attitude control system filter uses a constant integration time and that all of the samples of ES.Scan.CorrParms.Beta are the same. [In this case there will be only one set of ES.Scan.CorrParms.Beta correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the attitude control uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Scan.CorrParms.Beta describe the error correlation of a net LOS measure that includes the effect of the orientation of the sensor itself and the steering system that directs the beam optical axis relative to the sensor. While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of cosine Beta is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.Beta(j,1) = *a*, the large time asymptotic value

ES.Scan.CorrParms.Beta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.Beta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.Beta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.Beta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.Beta(j,k) where

index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the autocorrelation function for geometric errors in this CMMD is the same for all six degrees of freedom for the sensor (three sensor orientation and three sensor position errors.) Thus the description here follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_ ES.Scan.CorrParms.Alpha(j,k) ES.Swath.CorrParms.Beta(j)

20.2.45 ES.Scan.CorrParms.Gamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.Gamma(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the net rotation about the LOS at the sensor.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

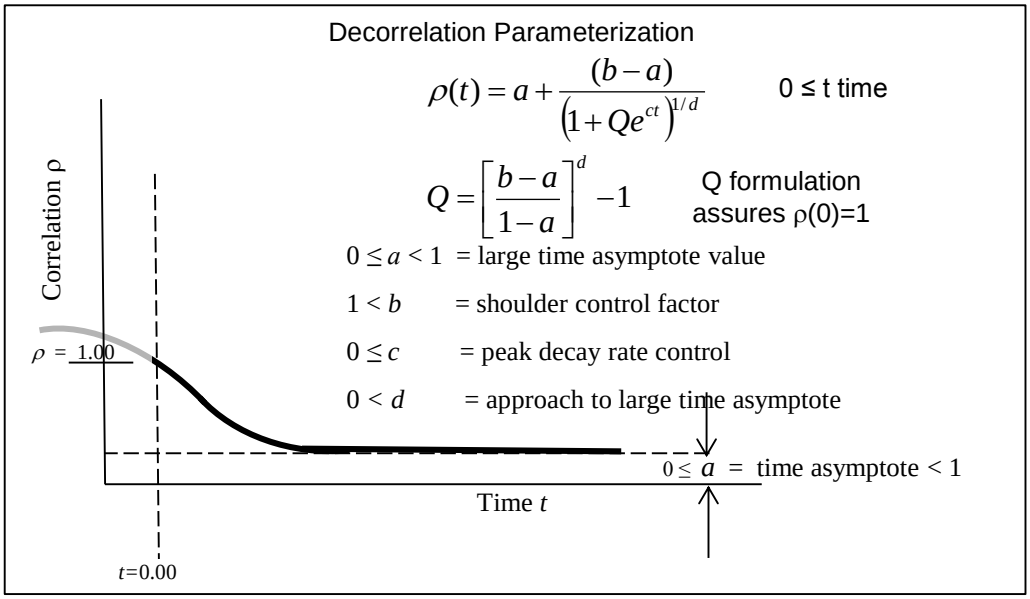
Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.Gamma(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor rotation about the line of sight as determined at each moment of sampling the error covariance matrix. A more complete description of the look axis geometric parameters is available in ES.Scan.SigmaGamma(j), SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), and in 10.1.2.3. The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other

processing yet introduced that could make unbiased errors correlated over longer time intervals.
It may be that the attitude control system filter uses a constant integration time and that all of the samples of ES.Scan.CorrParms.Gamma are the same. [In this case there will be only one set of ES.Scan.CorrTimeGamma correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the attitude control uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Scan.CorrParms.Gamma describe the error correlation of a net LOS measure that includes the effect of the orientation of the sensor itself and the steering system that directs the beam optical axis relative to the sensor. While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of Gamma is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.Gamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.Gamma(j,2= *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.Gamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.Gamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.Gamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.Gamma(j,k)

where index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the autocorrelation function for geometric errors in this CMMD is the same for all six degrees of freedom for the sensor (three sensor orientation and three sensor position errors.) Thus the description here follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.Gamma(j) 10.1.2.2 ES.Scan.CorrParms.Alpha(j,k)

20.2.46 ES.Scan.CorrParms.X(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.X(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the net sensor position x-coordinate in the earth frame.

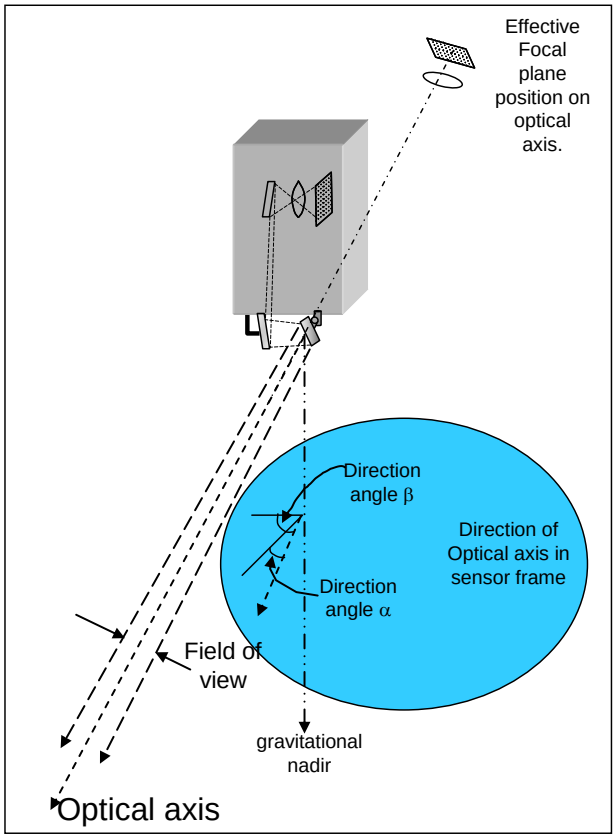
indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.X(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor x-position as determined at each moment of sampling the error covariance matrix. The ‘ES.Scan’ prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals.
It may be that the navigation system filter uses a constant integration time and

that all of the samples of ES.Scan.CorrParms.X are the same. [In this case there will be only one set of ES.Scan.CorrParms.X correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the navigation system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

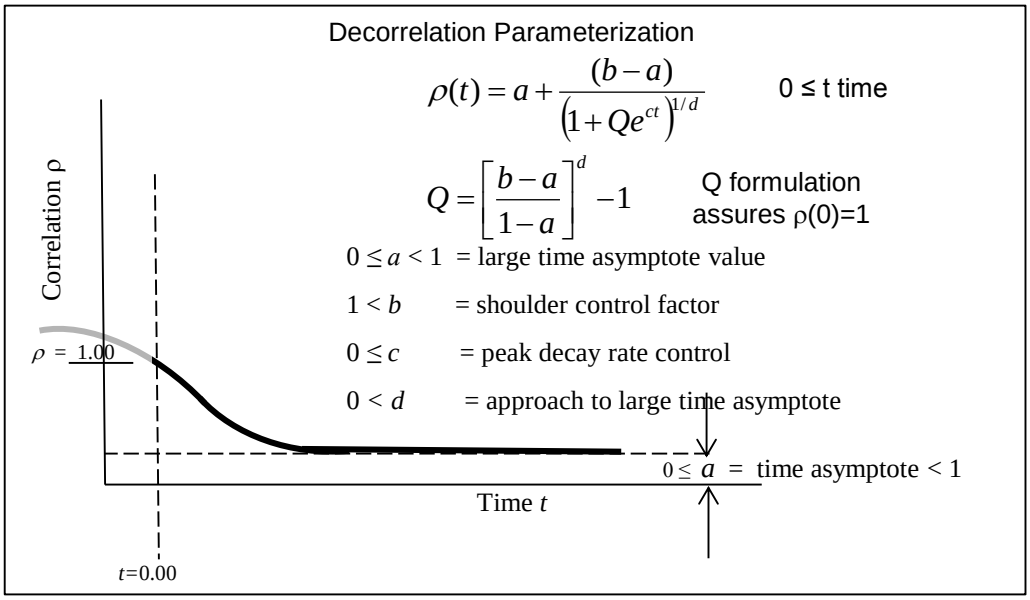
The quantities given by ES.Scan.CorrParms.X describe the error correlation of a net sensor position measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor. While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations.



The auto correlation of X is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time *t* is in seconds and *Q* is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.X(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.X(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.X(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.X(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.X(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \geq 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.X(j,k) where index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the autocorrelation function for geometric errors in this CMMD is the same for all six degrees of freedom for the sensor (three sensor orientation and three sensor position errors.) Thus the description here follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.X(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.47 ES.Scan.CorrParms.Y(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParms.Y(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the net sensor position y-coordinate in the earth frame.

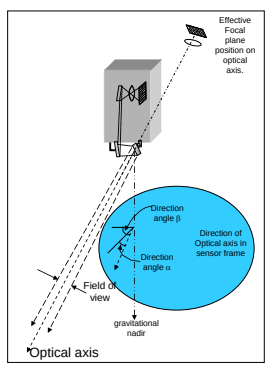
indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the

degree to which these errors are correlated over time.
ES.Scan.CorrParms.Y(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor y-position as determined at each moment of sampling the error covariance matrix. The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals.
It may be that the navigation system filter uses a constant integration time and that all of the samples of ES.Scan.CorrParms.Y are the same. [In this case there will be only one set of ES.Scan.CorrParms.Y correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the navigation system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

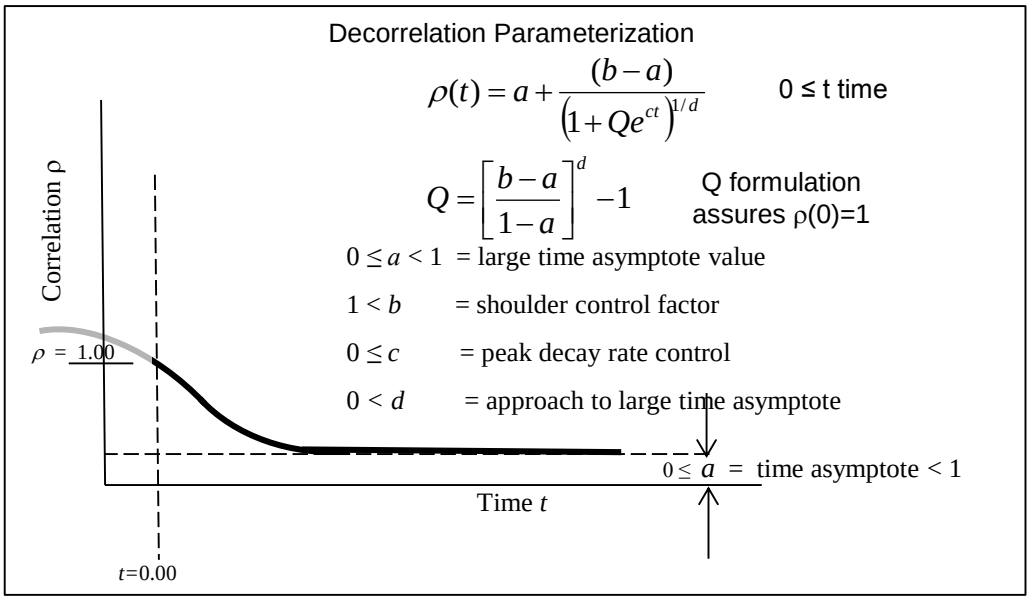
The quantities given by ES.Scan.CorrParms.Y describe the error correlation of a net sensor position measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.



While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of Y is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.Y(j,1= *a*, the large time asymptotic value.

ES.Scan.CorrParms.Y(j,2= *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.Y(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.Y(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.Y(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots,

j, the correlation decay is described in ES.Scan.CorrParmsT.Y(j,k) where index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the autocorrelation function for geometric errors in this CMMD is the same for all six degrees of freedom for the sensor (three sensor orientation and three sensor position errors.) Thus the description here follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.Y(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.48 ES.Scan.CorrParms.Z(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.Z(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the net sensor position z-coordinate in the earth frame.

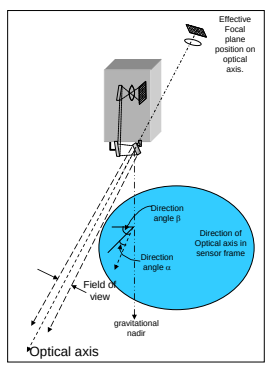
indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.Z(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor z-position as determined at each

moment of sampling the error covariance matrix. The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals.
It may be that the navigation system filter uses a constant integration time and that all of the samples of ES.Scan.CorrParms.Z are the same. [In this case there will be only one set of ES.Scan.CorrParms.Z correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the navigation system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

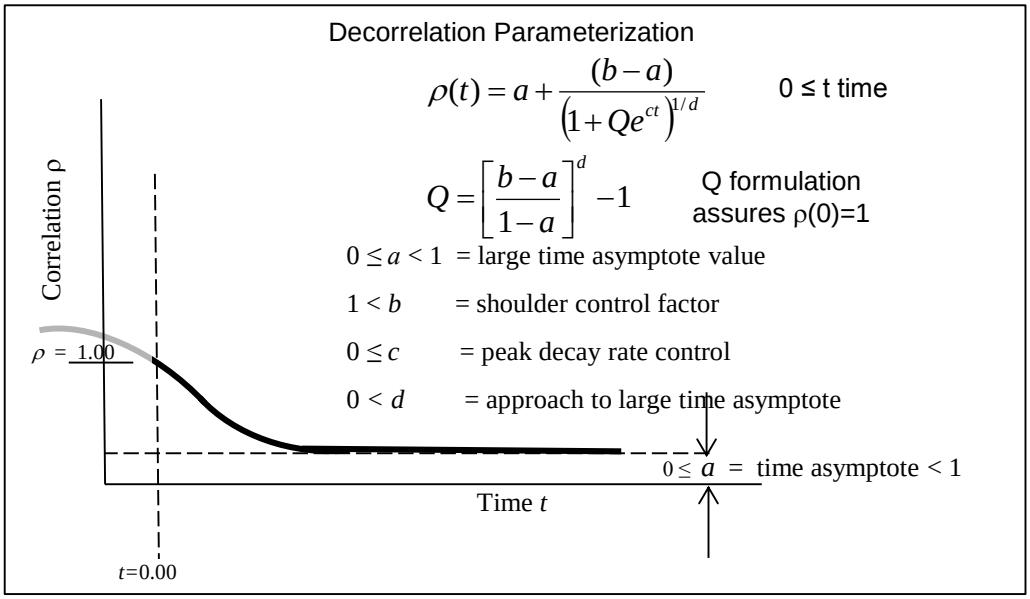
The quantities given by ES.Scan.CorrParms.Z describe the error correlation of a net sensor position measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.



While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of Z is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.Z(j,1= *a*, the large time asymptotic value.

ES.Scan.CorrParms.Z(j,2= *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.Z(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.Z(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.Z(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \geq 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.Z(j,k) where

index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the autocorrelation function for geometric errors in this CMMD is the same for all six degrees of freedom for the sensor (three sensor orientation and three sensor position errors.) Thus the description here follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.Z(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.49 ES.Scan.CorrParms.R(j,n,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.R(j,n,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error in the range from sensor to a lidar point.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.
Index, n, equals either [0,1] or [1,2] depending on the conventions of the implementing software code. The lower value of n carries data for ranging errors that do not grow with the range and the high value carries data that corresponds to an error that is proportional to the range itself.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, not only is the uncertainty in the data coming out of the navigation and steering/attitude control system required, but the range uncertainty, itself, is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time. ES.Scan.CorrParms.R(j,n,k) are used to compute the level of correlation over time for the error in the earth frame sensor to lidar point range as determined at each moment of sampling

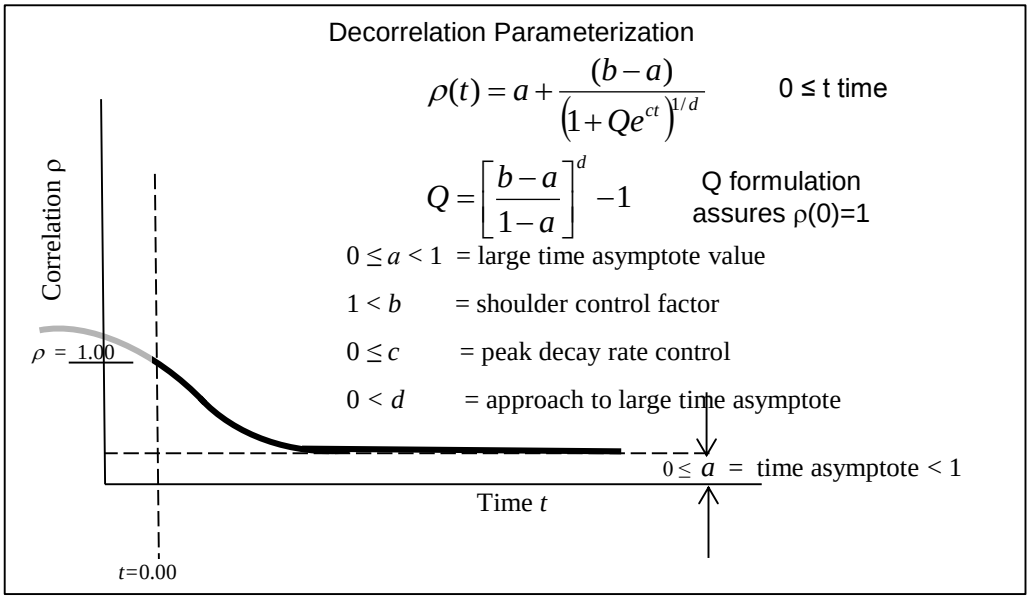
the error covariance matrix. The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals.
It may be that the atmospheric model is constant for a collect so that all of the (j indexed) samples of ES.Scan.CorrParms.R(j,n,k) are the same. [In this case there will be only one set of ES.Scan.CorrParms.R correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the lidar system tracks atmospheric model parameters and thus the ranging uncertainty could vary over a data collection.

The quantities given by ES.Scan.CorrParms.R describe the error correlation of a net ranging error resulting from applying the impact of refractive index estimation on 'rays' and round trip timing error contributions that are separated into path proportional and simply random error in estimating the ends of the roundtrip timing interval.

While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of R is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.R(j,n,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.R(j,n,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.R(j,n,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.R(j,n,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.R(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.R(j,k) where

index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the autocorrelation function for ranging errors in this CMMD is the same as for all six degrees of freedom for the sensor position. Thus, the prescription for calculating error correlations for points collected in different covariance snapshots follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.R(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.50 ES.Scan.CorrParms.AlphaBeta(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.AlphaBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and beta in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

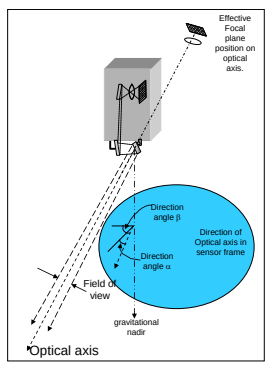
Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.AlphaBeta(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor orientation angles alpha and beta as determined at each moment of sampling the error covariance matrix.
The time dependence of cross correlation defined as

$$\rho_{\alpha\beta}(t = t_2 - t_1) = \left\langle \left(\alpha_{t_1} - \mu_{\alpha_{t_1}}\right)\left(\beta_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{a_{t_1}} \sigma_{\beta_{t_2}}\right)$$

The 'ES.Scan' prefix indicates that this is data at the stage of scan aggregation associated with there being no impacts of registration or other processing yet introduced that could make unbiased errors correlated over longer time intervals.

It may be that the navigation system filter uses a constant integration time and that all of the samples of ES.Scan.CorrParms.AlphaBeta are the same. [In this case there will be only one set of ES.Scan.CorrParms.AlphaBeta correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the navigation system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Scan.CorrParms.AlphaBeta describe the error cross correlation of a net sensor orientation measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.

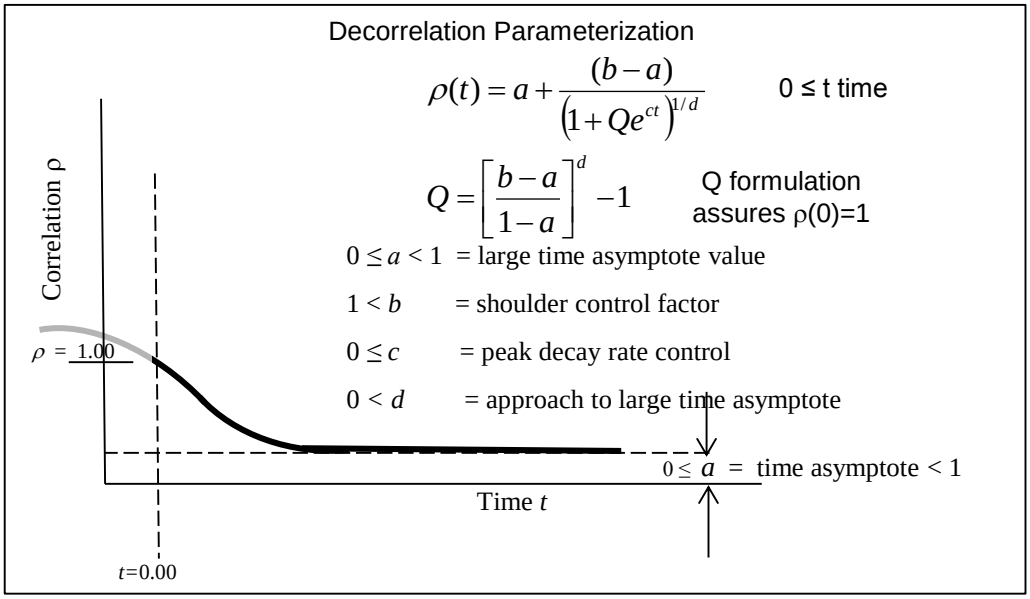


While the system's correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The cross correlation of alpha and beta as a function of time between measurements within a single covariance snapshot is modeled, like the autocorrelations, according to

$$\rho(t) = a + \frac{b - a}{\left(1 + Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per

the following diagram:



For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.AlphaBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.AlphaBeta(j,2= *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.AlphaBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.AlphaBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.AlphaBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Scan.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

On the other hand, if the times t_1 and t_2 belong to different time snapshots, j, the correlation decay is described in ES.Scan.CorrParmsT.AlphaBeta(j,k) where index, j, is index of the first snapshot in a consecutive pair of snapshots.

The parametric representation of the cross correlation function for geometric errors in this CMMD is the same as for the autocorrelations. Thus the description here follows that of ES.Scan.CorrParms.Alpha(j,k). See that definition paragraph for the procedure for dealing with the error cross correlation between points that have been collected during time intervals corresponding to different snapshots of the error covariance data.

If ES.Scan.CorrParms.AlphaBeta(j,k) is not populated, the cross correlation of alpha and beta is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.AlphaBeta(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.51 ES.Scan.CorrParms.AlphaGamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.AlphaGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.AlphaGamma(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor orientation angles alpha and gamma as determined at each moment of sampling the error

covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{\alpha\gamma}(t = t_2 - t_1) = \left\langle \left(\alpha_{t_1} - \mu_{\alpha_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{a_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Scan.CorrParms.AlphaGamma(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Scan' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParms.AlphaGamma(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParms.AlphaGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.AlphaGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParms.AlphaGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParms.AlphaGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.AlphaGamma(j,k) is not populated, the cross correlation of alpha and gamma is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.AlphaGamma(j)

ES.Scan.CorrParms.Alpha(j,k)

20.2.52 ES.Scan.CorrParms.BetaGamma(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParms.BetaGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations beta and gamma in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation

Variable, single float

Treatment/usage

The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.

ES.Scan.CorrParms.BetaGamma(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor orientation angles Beta and gamma as determined at each moment of sampling the error covariance matrix.

The time dependent cross correlation is defined as

$$\rho_{\beta\gamma}(t = t_2 - t_1) = \left\langle \left(\beta_{t_1} - \mu_{\beta_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{\beta_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Scan.CorrParms.BetaGamma(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.BetaGamma(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParms.BetaGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.BetaGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParms.BetaGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParms.BetaGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.BetaGamma(j,k) is not populated, the cross correlation of beta and gamma is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.BetaGamma(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.53 ES.Scan.CorrParms.XY(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParms.XY(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and y in the earth frame.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of

the degree to which these errors are correlated over time.
ES.Scan.CorrParms.XY(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coorldinates x and y as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{xy}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(y_{t_2} - \mu_{y_{t_2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{y_{t_2}}\right)$$

The description of ES.Scan.CorrParms.XY(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Scan' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParms.XY(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParms.XY(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.XY(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParms.XY(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParms.XY(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.XY(j,k) is not populated, the cross correlation of xand y is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics	
See Also	PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.XY(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.54 ES.Scan.CorrParms.XZ(j,k) [L1,2,3,4,5]

Brief Definition	ES.Scan.CorrParms.XZ(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and z in the earth frame.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty. Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.
Units/representation	Variable, single float
Treatment/usage	The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time. ES.Scan.CorrParms.XZ(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor coordinates x and z as determined at each moment of sampling the error covariance matrix. The time dependent cross correlation is defined as $\rho_{xz}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(z_{t_2} - \mu z_{t_2}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{z_{t_2}}\right)$ The description of ES.Scan.CorrParms.XZ(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of : • the ‘ES.Scan’ prefix • case that all the snapshots would have the same values of affected parameters • what physical effects are included in the calculation of this time dependent cross correlation • Significance of the parameters of the represented cross correlation • Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.XZ(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.XZ(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.XZ(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.XZ(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.XZ(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.XZ(j,k) is not populated, the cross correlation of x and z is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.XZ(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.55 ES.Scan.CorrParms.XAlpha(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParms.XAlpha(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation alpha in the earth frame.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error

propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.XAlpha(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor positioni coordinate x and angle alpha as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{x\alpha}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(\alpha_{t_2} - \mu_{\alpha_{t_2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{\alpha_{t_2}}\right)$$

The description of ES.Scan.CorrParms.XAlpha(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParms.XAlpha(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParms.XAlpha(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.XAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParms.XAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParms.XAlpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.XAlpha(j,k) is not populated, the cross correlation of x

and alpha is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.XAlpha(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.56 ES.Scan.CorrParms.XBeta(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.XBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation beta in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.XBeta(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate x and angle beta as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{x\beta}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(\beta_{t_2} - \mu_{\beta_{t2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{\beta_{t2}}\right)$$

The description of ES.Scan.CorrParms.XBeta(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time

dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.XBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.XBeta(j,2= *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.XBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.XBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.XBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.XBeta(j,k) is not populated, the cross correlation of x and beta is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.XBeta(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.57 ES.Scan.CorrParms.XGamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.XGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.

ES.Scan.CorrParms.XGamma(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate x and angle gamma as determined at each moment of sampling the error covariance matrix.

The time dependent cross correlation is defined as

$$\rho_{x\gamma}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Scan.CorrParms.XGamma(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.XGamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.XGamma(j,2= *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.XGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.XGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.XGamma(j,5) supplies the time in seconds over which

the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.XGamma(j,k) is not populated, the cross correlation of x and gamma is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.XGamma(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.58 ES.Scan.CorrParms.YZ(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.YZ(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates y and z in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.YZ(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinates y and z as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{yz}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(z_{t_2} - \mu_{z_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{z_{t_2}}\right)$$

The description of ES.Scan.CorrParms.YZ(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix

- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.YZ(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.YZ(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.YZ(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.YZ(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.YZ(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.YZ(j,k) is not populated, the cross correlation of y and z is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.YZ(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.59 ES.Scan.CorrParms.YAlpha(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.YAlpha(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle alpha in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation

Variable, single float

Treatment/usage

The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.YAlpha(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate y and orientation angle alpha as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{y\alpha}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(\alpha_{t_2} - \mu_{\alpha_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{\alpha_{t_2}}\right)$$

The description of ES.Scan.CorrParms.YAlpha(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParms.YAlpha(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParms.YAlpha(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.YAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.YAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.YAlpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.YAlpha(j,k) is not populated, the cross correlation of y and alpha is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.YAlpha(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.60 ES.Scan.CorrParms.YBeta(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParms.YBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle beta in the earth frame.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.YBeta(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate y and orientation angle beta as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{y\beta}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(\beta_{t_2} - \mu_{\beta_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{\beta_{t_2}}\right)$$

The description of ES.Scan.CorrParms.YBeta(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Scan' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.YBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.YBeta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.YBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.YBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.YBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.YBeta(j,k) is not populated, the cross correlation of y and beta is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.YBeta(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.61 ES.Scan.CorrParms.YGamma(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParms.YGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle gamma in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation

Variable, single float

Treatment/usage

The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.

ES.Scan.CorrParms.YGamma(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate y and orientation angle gamma as determined at each moment of sampling the error covariance matrix.

The time dependent cross correlation is defined as

$$\rho_{y\gamma}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Scan.CorrParms.YGamma(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.YGamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.YGamma(j,2) = *b*, the “shoulder” control factor which

must be greater than unity. This determines the early correlation decay rate second derivative

ES.Scan.CorrParms.YGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.YGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.YGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.YGamma(j,k) is not populated, the cross correlation of y and gamma is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.YGamma(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.62 ES.Scan.CorrParms.ZAlpha(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParms.ZAlpha(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle alpha in the earth frame.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.ZAlpha(j,k) are used to compute the level of cross

correlation over time for the errors in the earth frame sensor position coordinate z and orientation angle alpha as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{z\alpha}(t = t_2 - t_1) = \left\langle \left(z_{t_1} - \mu_{z_{t_1}}\right)\left(\alpha_{t_2} - \mu_{\alpha_{t2}}\right)\right\rangle / \left(\sigma_{z_{t_1}} \sigma_{\alpha_{t2}}\right)$$

The description of ES.Scan.CorrParms.ZAlpha(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Scan' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParms.ZAlpha(j,1= *a*, the large time asymptotic value.
ES.Scan.CorrParms.ZAlpha(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.ZAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParms.ZAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParms.ZAlpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.ZAlpha(j,k) is not populated, the cross correlation of z and alpha is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality

metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.ZAlpha(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.63 ES.Scan.CorrParms.ZBeta(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParms.ZBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle beta in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.ZBeta(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate z and orientation angle beta as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{z\beta}(t = t_2 - t_1) = \left\langle \left(z_{t_1} - \mu_{z_{t_1}}\right)\left(\beta_{t_2} - \mu_{\beta_{t2}}\right)\right\rangle / \left(\sigma_{z_{t_1}} \sigma_{\beta_{t2}}\right)$$

The description of ES.Scan.CorrParms.ZBeta(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParms.ZBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParms.ZBeta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParms.ZBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParms.ZBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParms.ZBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.ZBeta(j,k) is not populated, the cross correlation of z and beta is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.ZBeta(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.64 ES.Scan.CorrParms.ZGamma(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParms.ZGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error intra-snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle gamma in the earth frame.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error

propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Scan.CorrParms.ZGamma(j,k) are used to compute the level of cross correlation over time for the errors in the earth frame sensor position coordinate z and orientation angle gamma as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{z\gamma}(t = t_2 - t_1) = \left\langle \left(z_{t_1} - \mu_{z_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{z_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Scan.CorrParms.ZGamma(j,k) follows in close analogy to the description of ES.Scan.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Scan’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParms.ZGamma(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParms.ZGamma(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParms.ZGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParms.ZGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParms.ZGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

If ES.Scan.CorrParms.ZGamma(j,k) is not populated, the cross correlation of

z and gamma is to be treated as time independent.

All error covariance related data will be defined at the swath level too. There the impacts of processing and re-filtering the data will be represented.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Swath.CorrParms.ZGamma(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.65 ES.Scan.CorrParmsT.Alpha(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.Alpha(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal x-reference direction at the sensor.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.Alpha(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data

measurements.

- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.Alpha(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.Alpha(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.Alpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.Alpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.Alpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j)
ES.Swath.CorrParms.Alpha(j) ES.Scan.CorrParms.Alpha(j,k)

20.2.66 ES.Scan.CorrParmsT.Beta(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParmsT.Beta(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal y-reference direction at the sensor.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation

Variable, single float

Treatment/usage

As explained in the definition section for ES.Scan.CorrParms.Beta(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different

snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.Beta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT. Beta (j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT. Beta (j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT. Beta (j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT. Beta (j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaBeta (j) ES.Scan.CorrParms.Beta(j,k)

20.2.67 ES.Scan.CorrParmsT.Gamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.Gamma(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net rotation about the LOS at the sensor.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.Gamma(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of

"snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.Gamma (j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.Gamma (j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.Gamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.Gamma (j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.Gamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in

such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaGamma(j) ES.Scan.CorrParms.Gamma(j,k)

20.2.68 ES.Scan.CorrParmsT.X(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.X(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net sensor position x-coordinate in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.X(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to

asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.X(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.X(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.X(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.X(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.X(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting

points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaX(j) ES.Scan.CorrParms.X(j,k)

20.2.69 ES.Scan.CorrParmsT.Y(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParmsT.Y(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net sensor position y-coordinate in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation

Variable, single float

Treatment/usage

As explained in the definition section for ES.Scan.CorrParms.Y(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and

to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.Y(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.Y(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.Y(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.Y(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.Y(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi

stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaY(j) ES.Scan.CorrParms.Y(j,k)

20.2.70 ES.Scan.CorrParmsT.Z(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.Z(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the net sensor position z-coordinate in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.Z(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of “snapshots”, each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such

snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParmsT.Z(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.Z(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.Z(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.Z(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.Z(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics	
See Also	PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaZ(j) ES.Scan.CorrParms.Z(j,k)

20.2.71 ES.Scan.CorrParmsT.R(j,k) [L1,2,3,4,5]

Brief Definition	ES.Scan.CorrParmsT.R(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error in the range from sensor to a lidar point.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty. Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.
Units/representation	Variable, single float
Treatment/usage	As explained in the definition section for ES.Scan.CorrParms.R(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation. There are two other cases for point pair selection: • The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements. • The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values. The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.R(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.R(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.R(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.R(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.R(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-

snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaNbar(j) ES.Scan.CorrParms.R(j,k)

20.2.72 ES.Scan.CorrParmsT.AlphaBeta(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParmsT.AlphaBeta(j,k) gives the values of the parameters of the pre-processing geopositioning inter-snapshot error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and beta in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.AlphaBeta(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a transition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that

snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.AlphaBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.AlphaBeta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.AlphaBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.AlphaBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.AlphaBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the

asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.
**
If ES.Scan.CorrParmsT.AlphaBeta(j,k) is not populated, the cross correlation of alpha and beta is to be treated as time independent.

Governing quality metrics

See Also: PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j) ES.Scan.CorrParms.AlphaBeta(j,k)

20.2.73 ES.Scan.CorrParmsT.AlphaGamma(j,k) [L1,2,3,4,5]

Brief Definition: ES.Scan.CorrParmsT.AlphaGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and gamma in the earth frame.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation: Variable, single float

Treatment/usage: As explained in the definition section for ES.Scan.CorrParms.AlphaGamma(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of “snapshots”, each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParmsT.AlphaGamma(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParmsT.AlphaGamma(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation

decay rate second derivative.

ES.Scan.CorrParmsT.AlphaGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.AlphaGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.AlphaGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j) ES.Swath.CorrParms.Alpha(j)

20.2.74 ES.Scan.CorrParmsT.BetaGamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.BetaGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor orientations beta and gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.BetaGamma(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the

snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.BetaGamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.BetaGamma(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.BetaGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.BetaGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.BetaGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The

minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaBeta(j) ES.Scan.CorrParms.BetaGamma(j,k)

20.2.75 ES.Scan.CorrParmsT.XY(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.XY(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and y in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage

As explained in the definition section for ES.Scan.CorrParms.XY(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for

the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.XY(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.XY(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.XY(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.XY(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.XY(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaX(j) ES.Scan.CorrParms.XY(j,k)

20.2.76 ES.Scan.CorrParmsT.XZ(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.XZ(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and z in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.XZ(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of “snapshots”, each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data

measurements.

- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.XZ(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.XZ(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.XZ(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.XZ(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.XZ(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaX(j) ES.Scan.CorrParms.XZ(j,k)

20.2.77 ES.Scan.CorrParmsT.XAlpha(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParmsT.XAlpha(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation alpha in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation

Variable, single float

Treatment/usage

As explained in the definition section for ES.Scan.CorrParms.XAlpha(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different

snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.XAlpha(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.XAlpha(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.XAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.XAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.XAlpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j) ES.Scan.CorrParms.XAlpha(j,k)

20.2.78 ES.Scan.CorrParmsT.XBeta(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.XBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation beta in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.XBeta(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection

operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay

function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.XBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.XBeta (j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.XBeta (j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT XBeta (j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.XBet(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as

zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaX(j) ES.Scan.CorrParms.XBeta(j,k)

20.2.79 ES.Scan.CorrParmsT.XGamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.XGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.XGamma(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of “snapshots”, each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of

measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.XGamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.XGamma(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.XGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.XGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.XGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its

decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaX(j) ES.Scan.CorrParms.XGamma(j)

20.2.80 ES.Scan.CorrParmsT.YZ(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.YZ(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position

coordinates y and z in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation

Variable, single float

Treatment/usage

As explained in the definition section for ES.Scan.CorrParms.YZ(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely

resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.YZ(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.YZ(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.YZ(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.YZ(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.YZ(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters

selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaY(j) ES.Scan.CorrParms.YZ(j,k)

20.2.81 ES.Scan.CorrParmsT.YAlpha(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.YAlpha(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle alpha in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.YAlpha(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of “snapshots”, each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from

a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.YAlpha(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParmsT.YAlpha(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParmsT.YAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParmsT.YAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParmsT.YAlpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A

model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j) ES.Scan.CorrParms.YAlpha(j,k)

20.2.82 ES.Scan.CorrParmsT.YBeta(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.YBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle beta in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.YBeta(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region

is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:
ES.Scan.CorrParmsT.YBeta(j,1) = *a*, the large time asymptotic value.
ES.Scan.CorrParmsT.YBeta(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Scan.CorrParmsT.YBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Scan.CorrParmsT.YBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.
ES.Scan.CorrParmsT.YBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation

function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaBeta(j) ES.Scan.CorrParms.YBeta(j,k)

20.2.83 ES.Scan.CorrParmsT.YGamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.YGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation
Variable, single float

Treatment/usage
As explained in the definition section for ES.Scan.CorrParms.YGamma(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for

points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.YGamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.YGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.YGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.YGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.YGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different

snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaY(j) ES.Scan.CorrParms.YGamma(j,k)

20.2.84 ES.Scan.CorrParmsT.ZAlpha(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.ZAlpha(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle alpha in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.ZAlpha(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of “snapshots”, each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.ZAlpha(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.ZAlpha(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.ZAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.ZAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.ZAlpha(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also

PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaAlpha(j) ES.Scan.CorrParms.ZAlpha(j,k)

20.2.85 ES.Scan.CorrParmsT.ZBeta(j,k) [L1,2,3,4,5]

Brief Definition

ES.Scan.CorrParmsT.ZBeta(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle beta in the earth frame.

indices

Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation

Variable, single float

Treatment/usage

As explained in the definition section for ES.Scan.CorrParms.ZBeta(j,k), the CMMD data design does permit the case where the position and attitude determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.
In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-

sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.ZBeta(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.ZBeta(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.ZBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.ZBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.ZBeta(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0,

ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaBeta(j) ES.Scan.CorrParms.ZBeta(j,k)

20.2.86 ES.Scan.CorrParmsT.ZGamma(j,k) [L1,2,3,4,5]

Brief Definition ES.Scan.CorrParmsT.ZGamma(j,k) gives the values of the parameters of the pre-processing geopositioning error inter- snapshot correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1) minus 1, generally from 1 to one less than the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Scan(5) to PD.HiIndx.Scan(5), that is over the parameters of the scan level inter-snapshot error correlation model.

Units/representation Variable, single float

Treatment/usage As explained in the definition section for ES.Scan.CorrParms.ZGamma(j,k), the CMMD data design does permit the case where the position and attitude

determination system uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation. We support description of this case by allowing a number of "snapshots", each of which has a list of correlation decay model parameters describing intra-snapshot error correlation decay. Those are to be used whenever a pair of points subject to a relative measurement are selected from a single snapshot. The statistics for measurements within a single such snapshot are quasi-stationary, being a nearly stationary slice of a process which is not stationary for the entire collection operation.

There are two other cases for point pair selection:

- The point pair is selected, one each from consecutive snapshots, but do involve common support (filter input) for each of the data measurements.
- The pair comes from different snapshots and there is not common support. This arises when snapshot indices differ by more than one, or from consecutive snapshots but when sufficiently large time of measurement differences apply that the correlations have decayed to asymptotic values.

The first bullet above describes behavior in a **t**ransition region near the boundary of two covariance snapshots. The decay of correlation in this region is not necessarily the same as when both points share support within a single snapshot.

In the case of the first bullet, the correlation decay is represented by fitting the snaphot to snapshot correlation decay to models carried in ES.Scan.CorrParmsT.*. In none of these multi-snapshot cases can wide-sense stationarity be assumed. Subtracting a mean is not a single identical correction for all measurements in different parts of a snapshot or in different snapshots altogether. Consequently non-zero biases may be encountered leading to possible non-zero asymptotic error correlations.

The correlation decay parameterization in ES.Scan.CorrParmsT.* closely resembles that of ES.Scan.CorrParms.*. The functional form is the same, and to support the covariance matrix formalism, we insist that the correlation function be normalized to unity for zero lag. The difference in the CorrParms and CorrParmsT models is that CorrParms fits are made against data used for points whose time of measurement is in one snapshot using filter functions tuned only for the snapshot of the output points in that snapshot. CorrParmsT fits, on the other hand, are made against data that uses a filter optimized for the first snapshot for the first point and a filter optimized for the second snapshot for the second point. This will always have the behavior of a kind of

cross correlation since even for an element on the diagonal of the covariance matrix, different filters are being correlated to generate the correlation decay function.

For the case that PD.LoIndx.Scan(5)=1, the model parameters would be populated as follows:

ES.Scan.CorrParmsT.ZGamma(j,1) = *a*, the large time asymptotic value.

ES.Scan.CorrParmsT.ZGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Scan.CorrParmsT.ZGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Scan.CorrParmsT.ZGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

ES.Scan.CorrParmsT.ZGamma(j,5) supplies the time in seconds over which the intra-snapshot correlation in the jth snapshot experiences 99% of its decay toward the asymptotic value. This conveniently supports sorting points in groups with correlated vs uncorrelated errors.

It is also important to note that although the inter-snapshot correlation function can be evaluated for a difference in time measurements equal to zero, and the value of Q is based on the choice t=0, times for which an inter-snapshot correlation function can be validly used do NOT include t=0. The minimal value of the time difference between measurements for which ES.Scan.CorrParmsT.* is appropriate is one IPP. For t=0, ES.Scan.CorrParms.* would apply.

In the case of the second bullet, error for the selected points will have completely decayed correlations, with only residual biases able to result in any level of correlation at all. Point pairs that are neither in a single snapshot nor consecutive snapshots will be generated by using distinct filter parameters selected to minimize estimation errors in the presence of errors with the quasi stationary statistics of their respective snapshots. Thus, these filters will have no common input data. If they did, the snapshots would not constitute distinct quasistationary sample sets. For this case of points in different snapshots where the points do not both lie in the same transition zone, the safest strategy is to assume that the correlation is zero {as if the asymptotes were always zero} so as not to under estimate the relative errors.

Point pairs collected from adjacent snapshots but without common filter

support do not use the ES.Scan.CorrParmsT correlation decay parameterization. The error correlation for such a pair should be treated as zero despite the fact that could be an underestimate. The error correlation in such a case is not easily predictable within the current state of the art. The dataset does not contain a random distribution of errors. There is only one value of bias in snap shot one and a second value of bias in snapshot 2. A model for the correlation of such random biases remains to be devised.

Governing quality metrics

See Also PD.HiIndx.Scan ES.Scan.DelT_Cov ES.Scan.SigmaZ(j) ES.Scan.CorrParms.ZGamma(j,k)

20.2.87 ES.Scan.DelT_Cov [L1,2,3,4,5]

Brief Definition ES.Scan.DelT_Cov is the time increment between scan level error covariance data "snapshots"

indices none

Units/representation Seconds, float

Treatment/usage For this lidar data definition scheme, the pre-processing[121] error covariance data and decorrelation times derived from the attitude, navigation and sensor systems are reported on a uniform time grid. At each time hack or sampling time, the available sigmas, cross correlations and decorrelation times output as auxiliary information by the attitude and position determining system's software constitute a snap shot of the fundamental uncertainty data. The error propagation model defines the transformations required to convert such data to error ellipsoids on absolute and relative geometric measurements.

The time interval between reported samples of this data is given by ES.Scan.DelT_Cov. The value of ES.Scan.DelT_Cov is determined by the error propagation data support module in the product generating application so that meaningful variations in the error covariance data during a data set collection interval can be represented. The total number of "snap shots", Ns, of covariance data defines (Ns-1) sub-intervals of the total data set collection time is derived from PD.HiIndx.Scan(1).
The pre-processing error covariance at any particular time is determined by a suitable interpolation of the sequence of error covariance snap shots available at the ES.Scan.DelT_Cov intervals. The product ES.Scan.DelT_Cov × (Ns-

[121] "pre-processing" used in this context refers to values that apply when the process of creating a 3-D point cloud of measured positions has not introduced any adaptive corrections such as registrations or block adjustments that would tend to reduce geolocation errors from the native impacts of geometric projections necessary to "propagate" errors from sensor space to the product earth frame.

1) is equal to the total collection time for the data set. Thus, there will be a sample of error covariance representing the start time of collection and another sample of covariance representing the end time of collection. The application that populates ES.Scan.DelT_Cov must make the interval small enough that linear interpolation of every error covariance matrix element gives a fair representation (within 10%) of the instantaneous value of error covariance.

Governing quality metrics

See Also ES.Swath.DelT_Cov PD.HiIndx.Scan

20.2.88 ES.Scan.GM_SigmaR(j) [L1,2,3,4,5]

Brief Definition ES.Scan.GM_SigmaR(j) is the range independent range uncertainty at the scan level. (Geiger Mode only)

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Meters, float

Treatment/usage ES.Scan.GM_SigmaR(j) should only be populated for Geiger mode based data sets, as this item is not intended to account for the target dependent peak thresholding induced range uncertainties associated with linear mode returns.

ES.Scan.GM_SigmaR(j) represents range uncertainty at the scan level of data aggregation, that is, it is a pre-processing valuation of range uncertainty.[87]

Discussion of Geiger Mode range uncertainty:

The range uncertainty is the uncertainty in the product of the speed of light in the atmosphere (which we can treat as a ray path mean of the index of refraction times the free space constant, c) and the round trip time, τ_{rt}. Then the range independent range uncertainty is given by

$\Delta R = c\tau_{rt}\left(\bar{n}_{path}\frac{\Delta\tau_{rt}}{\tau_{rt}}\right)$ where we exclude terms that cause the ranging error to grow with range (error in the path averaged index of refraction and the error in the mean clock step duration). The round trip timing uncertainty we use here is that of ES.Swath.GM_SigmaT and includes the uncertainty only of identifying the moments of transmission and reception. The impact of the clock step error on the length of the round trip time interval measurement is not included here.

ES.Scan.GM_SigmaR(j) combines the effects of transmit and receive time uncertainties and time to range conversion. The propagation interval's

combined start and end time uncertainty alone is given by ES.Swath.GM_SigmaT. ES.Scan.GM_SigmaR(j) applies on a covariance "snapshot" level and expresses the consequence of the timing uncertainty in ES.Swath.GM_SigmaT as a range uncertainty. This means that the highly uncorrelated photon transmission and arrival effects are converted to a range based on a speed of light estimate good for a single covariance snapshot. Defining the data this way is convenient because we have an important range uncertainty available directly in the metadata and we can find the path averaged speed of light during a specific covariance "snapshot" interval by taking the ratio of ES.Scan SigmaR(j) to ES.Swath.GM_SigmaT. That speed of light, in turn, can be used to determine the uncertainty in range for range *dependent* error such as that due to the clock error that is reported in SC.Swath.SigmaClkStep.
The physical atmospheric parameters can be made available in PF.TimeToRange.* so that alternative propagation models can be used to determine the impacts of refraction in the atmosphere. If an alternative model is used to create an alternative product, there will likely be new values of ES.Scan.GM_SigmaR(j) and PF.TimeToRange.Model stored in the new dataset.

Note that ES.Scan.GM_SigmaR(j) tracks the covariance snapshots and represents the conversion of photon timing uncertainties to range uncertainties via slowly changing (i.e., environmental) sources of ranging error (refractive index fluctuation and path attenuations). High frequency shifts in the roundtrip time due to sensor induced attenuation variations and resulting Poisson distributed GM trigger time variations are addressed directly in ES.Swath.GM_Sigma_T. The high frequency range uncertainties add to the slowly varying range errors given by SC.Swath.SigmaClkStep and errors in estimating the index of refraction averaged over a ray. Thus, a combining procedure involves picking a snapshot interval for a particular pulse and building the range dependent ranging uncertainty from SC.Swath.SigmaClkStep, ES.Scan.SigmaNbar(j) and the path averaged speed of light ES.Scan SigmaR(j)/ES.Swath.GM_SigmaT. This can be RSS'd with ES.Scan.GM_SigmaR(j) for the same snapshot to find a total range uncertainty at a point.

This CMMD assumes there is no correlation between the ranging and platform position and orientation errors and does not provide data to convey range error cross correlations .

The above discussion is only about quantifying the *uncertainty* in derived ranges for lidar measurements. Evaluation of the **range** itself does require careful attention to the value of the index of refraction, the grazing angle and the altitude if valid ranges are to be derived. In the case of geometries subject to ray bending in addition to path dilation relative to vacuum propagation, the conversion of range to XYZ coordinates must include the impact of ray curvature to obtain the best achievable coordinate accuracy for single lidar position measurements.

Governing quality metrics

See Also ES.Swath.GM_Sigma_T(k) ES.Scan.SigmaNbar(j) SC.Swath.SigmaClkStep

20.2.89 ES.Scan.RhoAlphaGamma(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoAlphaGamma(j) is the scan level correlation between the uncertainty in cos(alpha) and the uncertainty in the angle gamma, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Dimensionless, float

Treatment/usage ES.Scan.RhoAlphaGamma(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.

ES.Scan.RhoAlphaGamma(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation between uncertainties in cosine(alpha) and uncertainties in gamma for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted

that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]

The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame).

The angle gamma, which is a subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.90 ES.Scan.RhoXBeta(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.RhoXBeta(j) is the scan level correlation between the uncertainties in the sensor x-position coordinate and cosine of angle beta, and is part of the sensor error covariance data.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: ES.Scan.RhoXBeta(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.

ES.Scan.RhoXBeta(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of x-position-cosine(beta) uncertainty correlation for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10%[122] when linearly interpolated to times different than that of the j index values.

[122] If this ever proves to a be a capability limiting convention, we could add a new metadata item that is the interpolation accuracy provided and have this paragraph point to that new parameter. Need is not sufficiently likely to add the item now.

In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame).
The x- position subject of this statistic is measured in the positive (increases to north for NED frame) horizontal x-direction in the earth frame at the sensor position at the time associated with index, j, value of the error covariance sample.

Governing quality metrics

See Also ES.Scan.DelT_Cov

20.2.91 ES.Scan.RhoAlphaBeta(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoAlphaBeta(j) is the scan level correlation between uncertainties in the cosines of the line of sight angles alpha and beta, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Dimensionless, float

Treatment/usage ES.Scan.RhoAlphaBeta(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Scan.RhoAlphaBeta(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of cosine(alpha)-cosine(beta) uncertainty correlation for

each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame).
The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) in the earth frame at the sensor position at the time associated with the index, j, value of the error covariance sample.

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.92 ES.Scan.RhoBetaGamma(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.RhoBetaGamma(j) is the scan level correlation between the uncertainty in cos(beta) and the uncertainty in the angle gamma, and is part of the sensor error covariance data.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: ES.Scan.RhoBetaGamma(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.

ES.Scan.RhoBetaGamma(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the correlation between uncertainties in cosine(alpha) and uncertainties in gamma for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame).
The angle gamma, a subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also ES.Scan.DelT_Cov PD.HiIndx.Scan(1)

20.2.93 ES.Scan.RhoXAlpha(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoXAlpha(j) is the scan level correlation between the sensor x-position coordinate uncertainty and the uncertainty in cosine of angle alpha, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Dimensionless, float

Treatment/usage ES.Scan.RhoXAlpha(j) is an off diagonal element of a matrix representing the error error covariance of the position, line of sight and range data. The

reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Scan.RhoXAlpha(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of x-position-cosine(alpha) uncertainty correlation for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame).
The x- position subject of this statistic is measured in the positive (increases to north for NED frame) horizontal x-direction in the earth frame at the sensor position at the time associated with index, j, value of the error covariance sample.

Governing quality metrics

See Also ES.Scan.DelT_Cov

20.2.94 ES.Scan.RhoXGamma(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoXGamma(j) is the scan level correlation between the sensor x-position coordinate uncertainty and the uncertainty in the angle gamma, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-

	processing position and LOS uncertainty
Units/representation	Dimensionless, float
Treatment/usage	ES.Scan.RhoXGamma(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Scan.RhoXGamma(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation for uncertiany in x-position and uncertainty in the angle, gamma, for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.] The x coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame). The angle gamma, whose uncertainty is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.
Governing quality metrics	
See Also	ES.Scan.DelT_Cov

20.2.95 ES.Scan.RhoXY(j) [L1,2,3,4,5]

Brief Definition	ES.Scan.RhoXY(j) is the scan level correlation between the uncertainty in the sensor x-position coordinate and uncertainty in the y-position coordinate, and is part of the sensor error covariance data.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	Dimensionless, float
Treatment/usage	ES.Scan.RhoXY(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Scan.RhoXY(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values correlation of in the uncertainties in x-position and y-position for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.] The x-coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame). The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame).
Governing quality metrics	
See Also	ES.Scan.DelT_Cov

20.2.96 ES.Scan.RhoXZ(j) [L1,2,3,4,5]

Brief Definition	ES.Scan.RhoXZ(j) is the scan level correlation between the uncertainty in the sensor x-position coordinate and the uncertainty in the z-position coordinate, and is part of the sensor error covariance data.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	Dimensionless, float
Treatment/usage	ES.Scan.RhoXZ(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Scan.RhoXZ(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation the uncertainty in x-position and uncertainty in z-position for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.] The x-coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame). The z-coordinate is measured along the vertical reference direction (i.e. down for a NED frame).
Governing quality	

metrics
See Also ES.Scan.DelT_Cov

20.2.97 ES.Scan.RhoYAlpha(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoYAlpha(j) is the scan level correlation between the y-position coordinate uncertainty and uncertainty in the cosine of angle alpha, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Dimensionless, float

Treatment/usage ES.Scan.RhoYAlpha(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.

ES.Scan.RhoYAlpha(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation in the uncertainty in y-position and uncertainty in cosine(alpha) for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]

The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame).

The y- position subject of this statistic is measured in the positive (increases

to east for NED frame) horizontal y-direction in the earth frame at the sensor position at the time associated with index, j, value of the error covariance sample.

Governing quality metrics

See Also ES.Scan.DelT_Cov

20.2.98 ES.Scan.RhoYBeta(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoYBeta(j) is the scan level correlation between the uncertainty in the y-position coordinate and uncertainty in the cosine of angle beta, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Dimensionless, float

Treatment/usage ES.Scan.RhoYBeta(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.

ES.Scan.RhoYBeta(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation in uncertainties for y-position and cosine(beta) for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]

The angle beta, whose cosine is the subject of this statistic, is measured from

the horizontal y-reference direction (i.e. east for a NED frame).
The y- position subject of this statistic is measured in the positive (increases to east for NED frame) horizontal y-direction in the earth frame at the sensor position at the time associated with index, j, value of the error covariance sample.

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.99 ES.Scan.RhoYGamma(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.RhoYGamma(j) is the scan level correlation between the uncertainty in the sensor y-position coordinate and uncertainity in the angle gamma, and is part of the sensor error covariance data.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: ES.Scan.RhoYGamma(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Scan.RhoYGamma(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation of uncertanties in y-position and angle gamma for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance

data is before the introduction of processing generated corrections.]
The y coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame).
The angle gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also ES.Scan.DelT_Cov

20.2.100 ES.Scan.RhoYZ(j) [L1,2,3,4,5]

Brief Definition ES.Scan.RhoYZ(j) is the scan level correlation between the uncertainty in the sensor y-position coordinate and uncertainty in the sensor z-position coordinate, and is part of the sensor error covariance data.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation Dimensionless, float

Treatment/usage ES.Scan.RhoYZ(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Scan.RhoYZ(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of y-position--z-position correlation for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance

	data is before the introduction of processing generated corrections.] The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame). The z-coordinate is measured along the vertical reference direction (i.e. down for a NED frame).
Governing quality metrics	
See Also	ES.Scan.DelT_Cov

20.2.101 ES.Scan.RhoZAlpha(j) [L1,2,3,4,5]

Brief Definition	ES.Scan.RhoZAlpha(j) is the scan level correlation between the uncertainty in the sensor z-position coordinate and the uncertainty in the cosine of angle alpha, and is part of the sensor covariance data.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	Dimensionless, float
Treatment/usage	ES.Scan.RhoZAlpha(j) is an off diagonal element of a matrix representing the covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Scan.RhoZAlpha(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation uncertainty in the sensor z-position and the uncertainty in cosine(alpha) for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the

case that we wish access to both the pre-correction covariance and the post-correction error covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The angle alpha, whose cosine is the subject of this statistic, is measured from the sensor horizontal x-reference direction (i.e. north for a NED frame).
The sensor z- position subject of this statistic is measured in the positive (increases downward for NED frame) vertical direction in the earth frame at the sensor position at the time associated with index, j, value of the error covariance sample.

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.102 ES.Scan.RhoZBeta(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.RhoZBeta(j) is the scan level correlation between uncertainty in the z-position coordinate and uncertainty in the cosine of angle beta, and is part of the sensor error covariance data.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: ES.Scan.RhoZBeta(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Scan.RhoZBeta(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation in the uncertainty of the sensor z-position and uncertainty in cosine(beta) for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line

of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The angle beta, whose cosine is the subject of this statistic, is measured from the sensor horizontal y-reference direction (i.e. east for a NED frame).
The sensor z- position subject of this statistic is measured in the positive (increases downward for NED frame) vertical direction in the earth frame at the sensor position at the time associated with index, j, value of the covariance sample.

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.103 ES.Scan.RhoZGamma(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.RhoZGamma(j) is the scan level correlation between the uncertainity in the sensor z-position coordinate and uncertainty in the angle, gamma, and is part of the sensor error covariance data.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: ES.Scan.RhoZGamma(j) is an off diagonal element of a matrix representing the error covariance of the position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Scan.RhoZGamma(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation between the uncertainty in the z-position and the uncertainty in the angle, gamma, for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the

scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
The z coordinate is measured along vertical reference direction (i.e. down for a NED frame).
The angle, gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.104 ES.Scan.SigmaAlpha(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.SigmaAlpha(j) describes the sensor orientation uncertainty and is given by the standard deviation of the cosine of angle alpha measurement, a contributor to the diagonal of the error covariance matrix for the sensor.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, single float

Treatment/usage: The angle, alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame) at the sensor to the receiver optical axis, that is, the line of sight. This angle is used only to support error propagation, and only its uncertainty is documented. For purposes of computing lidar point positions, the direction of the optical axis is defined as the cross product of the receive array ECEF axis unit vectors (see 10.1.2.2).
ES.Scan.SigmaAlpha(j) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance

matrix.
ES.Scan.SigmaAlpha(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the standard deviation of cosine(alpha) measurement for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]

Governing quality metrics

See Also: ES.Scan.DelT_Cov

20.2.105 ES.Scan.SigmaBeta(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.SigmaBeta(j) describes the sensor orientation uncertainty and is given by the standard deviation of the cosine of angle beta measurement, a contributor to the diagonal of the error covariance matrix for the sensor.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the covariance matrix for pre-processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: The angle, beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) at the sensor to the receiver optical axis, that is, the line of sight. This angle is used only to support error propagation, and only its uncertainty is documented. For purposes of computing lidar point positions, the direction of the optical axis is defined as the cross product of the receive array ECEF axis unit vectors (see

10.1.2.2).
ES.Scan.SigmaBeta(j) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.
ES.Scan.SigmaBeta(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the standard deviation of cosine(beta) measurement for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]

Governing quality metrics

See Also ES.Scan.DelT_Cov

20.2.106 ES.Scan.SigmaGamma(j) [L1,2,3,4,5]

Brief Definition ES.Scan.SigmaGamma(j) describes the sensor orientation uncertainty and is given by the standard deviation of the angle gamma measurement, a contributor to the diagonal of the error covariance matrix for the sensor.

indices Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-

processing position and LOS uncertainty

Units/representation: Dimensionless, float

Treatment/usage: ES.Scan.SigmaGamma(j) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.

ES.Scan.SigmaGamma(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of the standard deviation of gamma measurement for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]

The angle gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also: ES.Scan.DelT_Cov 10.1.2.2 SC.Pulse.ArrayMajorAxis.*TileNum*(ip,j)

20.2.107 ES.Scan.SigmaX(j) [L1,2,3,4,5]

Brief Definition: ES.Scan.SigmaX(j) describes the sensor position uncertainty and is given by the standard deviation of the x-coordinate measurement, a contributor to the

	diagonal of the error covariance matrix for the sensor.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	meters, single float
Treatment/usage	ES.Scan.SigmaX(j) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the covariance matrix. ES.Scan.SigmaX(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The x-coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame) at the sensor. The reported values of the standard deviation of the measurement of position coordinate, X, for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
Governing quality metrics	
See Also	ES.Scan.DelT_Cov

20.2.108 ES.Scan.SigmaY(j) [L1,2,3,4,5]

Brief Definition	ES.Scan.SigmaY(j) describes the sensor position uncertainty and is given by

	the standard deviation of the y-coordinate measurement, a contributor to the diagonal of the error covariance matrix for the sensor.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	meters, float
Treatment/usage	ES.Scan.SigmaY(j) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the covariance matrix. ES.Scan.SigmaY(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame) at the sensor. The reported values of the standard deviation of the measurement of position coordinate, Y, for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
Governing quality metrics	
See Also	ES.Scan.DelT_Cov

20.2.109 ES.Scan.SigmaZ(j) [L1,2,3,4,5]

Brief Definition	ES.Scan.SigmaZ(j) describes the sensor position uncertainty and is given by the standard deviation of the z-coordinate measurement, a contributor to the diagonal of the error covariance matrix for the sensor.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	meters, float
Treatment/usage	ES.Scan.SigmaZ(j) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the covariance matrix. ES.Scan.SigmaZ(j) represents a value that includes no impact of processing that might improve geometric fidelity, such as internal registrations or block adjustments. The z-coordinate is measured along the z-reference direction (i.e. down for a NED frame) at the sensor. The reported values of the standard deviation of the measurement of position coordinate, Z, for each value of index j, are successive time samples at intervals given by ES.Scan.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the scan level error covariance matrix is directly applicable to lidar points that have not been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Scan.xxx error covariance data is before the introduction of processing generated corrections.]
Governing quality metrics	
See Also	ES.Scan.DelT_Cov

20.2.110 ES.Scan.SigmaNbar(j) [L1,2,3,4,5]

Brief Definition	ES.Scan.SigmaNbar(j) describes the uncertainty in the path averaged index of refraction.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty
Units/representation	dimensionless, float
Treatment/usage	The conversion of round trip times to range involves scaling the time to distance via the speed of light averaged over the propagation path. The average index of refraction over a ray path ($\bar{n}$) can be used to determine the average speed of light on a ray path for performing this conversion. Any error in the index of refraction will cause an incorrect value of the speed of light to be used in the time to range conversion. ES.Scan.SigmaNbar(j) is the standard deviation of the distribution of path averaged refractive index estimates , that is, the uncertainty measure for $\bar{n}$ provided for each covariance snapshot, j. Since the range is proportional to $\bar{n}$, the range error due to errors in $\bar{n}$ is proportional to the range, as discussed in 10.5.1.3. Consequently, ES.Scan.SigmaNbar(j) is part of computing the range dependent range uncertainties.
Governing quality metrics	
See Also	SC.Swath.SigmaClkStep

ES.Swath.*

ES.Swath.* parameters describe swath level data, including adaptive processing based corrections. For intermediate products, this includes corrections reported up to the current product level.

20.2.111 ES.Swath.CorrParms.Alpha(j,k) [L2,3,4,5]

Brief Definition	ES.Swath.CorrParms.Alpha(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal x-reference direction at the sensor.
indices	Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.
Units/representatio	variable, single float

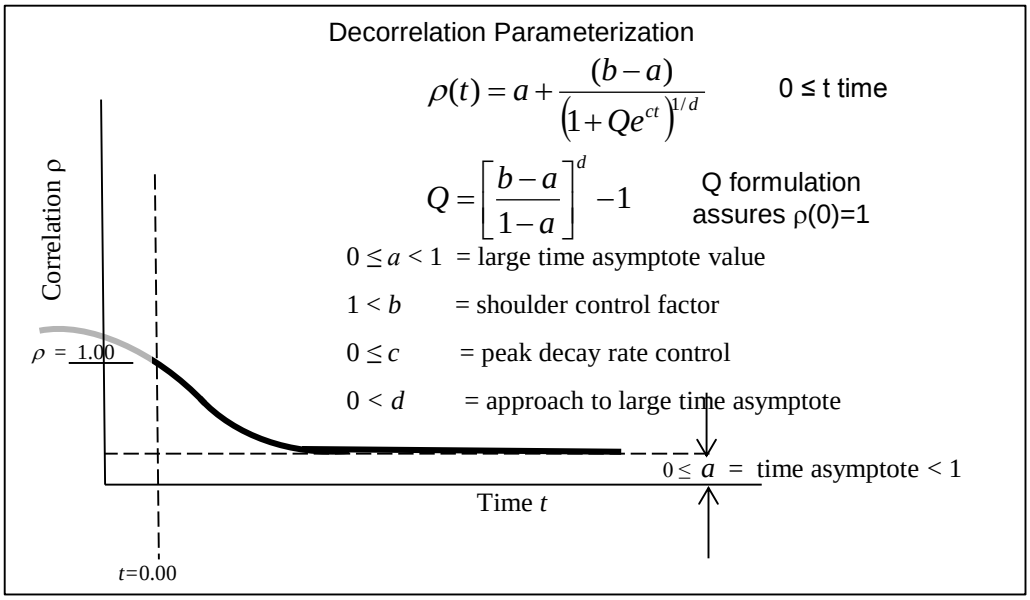
n
Treatment/usage

The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Swath.CorrParms.Alpha(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor x-axis based direction cosine as determined at each moment of sampling the post processing error covariance matrix. A more complete description of the look axis geometric parameters is available in ES.Swath.SigmaAlpha(j), SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), and in 10.1.2.3. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation associated with improvements due to registration or other processing that could make unbiased errors correlated over longer time intervals while reducing their magnitude due to adaptive or truth based geometric updates. Such processing is intended to reduce relative error in the swath level point cloud.
It may be that the processing registration and resampling filters effect a constant integration time and that all of the samples of ES.Swath.CorrParms.Alpha are the same. [In this case there will be only one set of ES.Swath.CorrParms.Alpha correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the adaptive remosaicking uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Swath.CorrParms.Alpha describe the error correlation of a net LOS measure that includes the residual effects of the orientation of the sensor itself and the steering system that directs the beam optical axis relative to the sensor. While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of cosine Alpha is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.Alpha(j,1) = a, the large time asymptotic value

ES.Swath.CorrParms.Alpha(j,2) = b, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.Alpha(j,3) = c, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.Alpha(j,4) = d, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter t of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the

correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of negligible numerical significance.

All error covariance related data will be defined at the scan level too. There, the impacts of processing and re-filtering the data will be omitted.

If ES.Swath.CorrParms.* is not populated, then the data must treated as not available and should not be taken to be zero.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.Alpha(j)

20.2.112 ES.Swath.CorrParms.Beta(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.Beta(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net direction cosine LOS component measured from the horizontal y-reference direction at the sensor.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation variable, single float

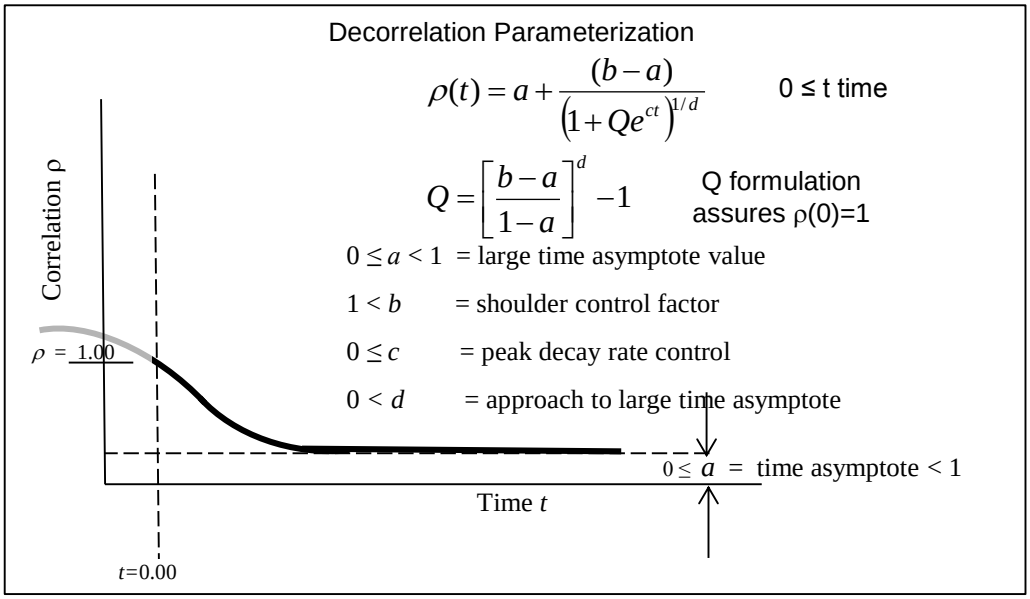
Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Swath.CorrParms.Beta(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor y-axis based direction cosine as determined at each moment of sampling the post processing error covariance matrix. A more complete description of the look axis geometric parameters is available in ES.Swath.SigmaBeta(j), SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), and in 10.1.2.3. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation

associated with improvements due to registration or other processing that could make unbiased errors correlated over longer time intervals while reducing their magnitude due to adaptive or truth based geometric updates. Such processing is intended to reduce relative error in the swath level point cloud.
It may be that the processing registration and resampling filters effect a constant integration time and that all of the samples of ES.Swath.CorrParms.Beta are the same. [In this case there will be only one set of ES.Swath.CorrParms.Beta correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the adaptive remosaicking uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Swath.CorrParms.Beta describe the error correlation of a net LOS measure that includes the residual effects of the orientation of the sensor itself and the steering system that directs the beam optical axis relative to the sensor. While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of cosine Beta is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.Beta(j,1) = *a*, the large time asymptotic value

ES.Swath.CorrParms.Beta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.Beta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.Beta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of

negligible numerical significance.

All error covariance related data will be defined at the scan level too. There, the impacts of processing and re-filtering the data will be omitted.

If ES.Swath.CorrParms.* is not populated, then the data must treated as not available and should not be taken to be zero.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.Beta(j)

20.2.113 ES.Swath.CorrParms.Gamma(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.Gamma(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net rotation about the LOS at the sensor.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the scan level error correlation model. Index k, ranges from

Units/representation Variable, single float

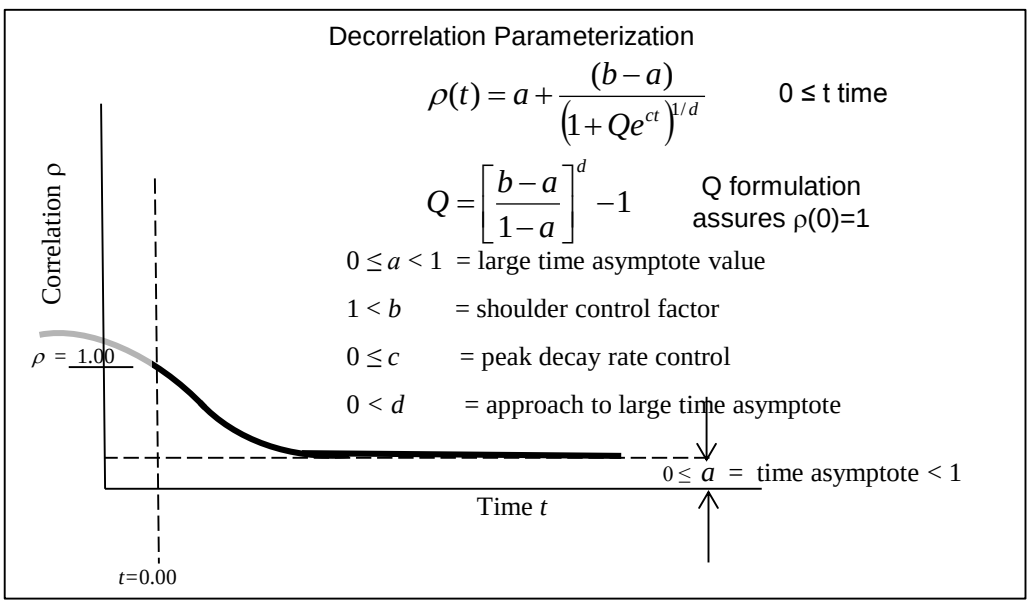
Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Swath.CorrParms.Gamma(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor rotation about the line of sight as determined at each moment of sampling the error covariance matrix. A more complete description of the look axis geometric parameters is available in ES.Swath.SigmaGamma(j), SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), and in 10.1.2.3. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation associated with impacts of registration or other processing introduced that could make unbiased errors correlated over longer time intervals.
It may be that the processing registration filter uses a constant integration time and that all of the samples of ES.Swath.CorrParms.Gamma are the same. [In this case there will be only one set of ES.Swath.CorrTimeGamma correlation

model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the attitude control uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Swath.CorrParms.Gamma describe the error correlation of a net LOS measure that includes the effect of the orientation of the sensor itself and the steering system that directs the beam optical axis relative to the sensor. While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of Gamma is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.Gamma(j,1) = *a*, the large time asymptotic value
ES.Swath.CorrParms.Gamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.Gamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.Gamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2-t_1) \geq 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of negligible numerical significance.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will be omitted.

If ES.Swath.CorrParms.* is not populated, then the data must treated as not available and should not be taken to be zero.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.Gamma(j) 10.1.2.2 ES.Swath.CorrParms.Alpha(j,k)

20.2.114 ES.Swath.CorrParms.X(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.X(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net sensor position x-coordinate in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-

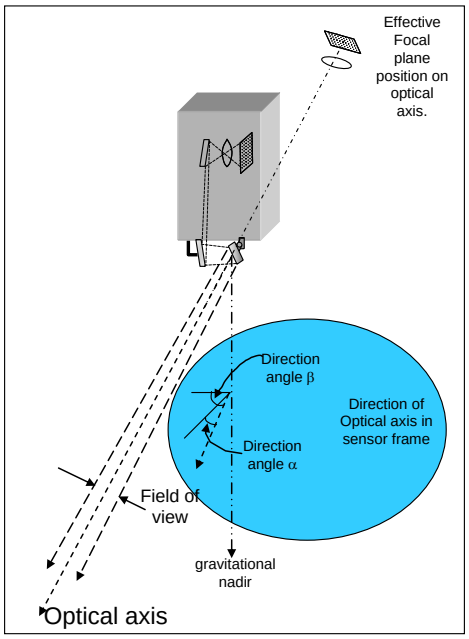
processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Swath.CorrParms.X(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor x-position as determined at each moment of sampling the error covariance matrix. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation associated with there being improvements due to registration or other processing[123] that could make errors correlated over longer time intervals.
It may be that the registration processing filter uses a constant integration time and that all of the samples of ES.Swath.CorrParms.X are the same. [In this case there will be only one set of ES.Swath.CorrParms.X correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the processing uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

The quantities given by ES.Swath.CorrParms.X describe the error correlation of a net sensor position measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.

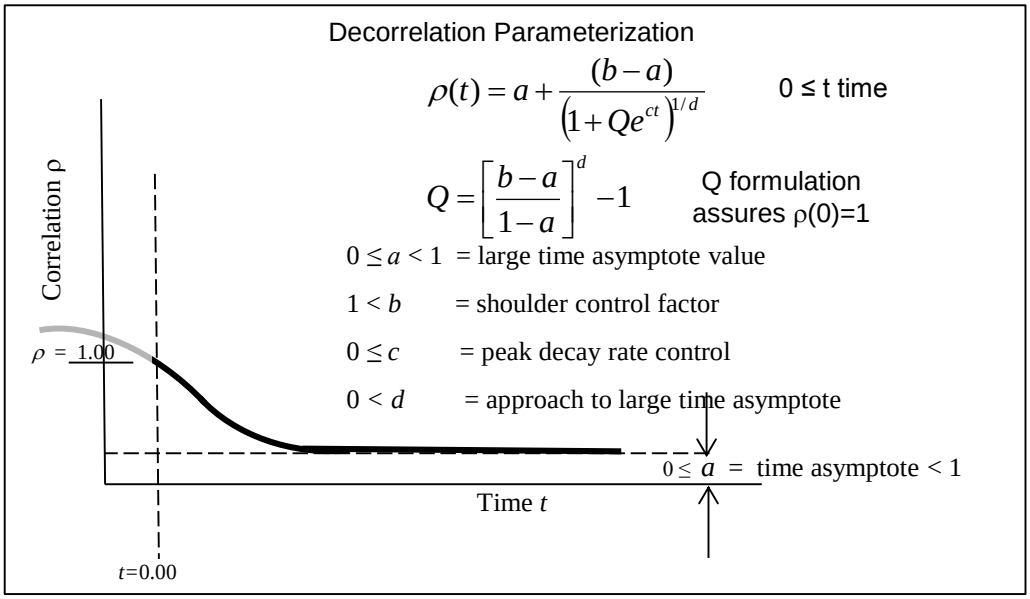
[123] So this is an *effective* correlation model parameter value set, as opposed to the mechanical/navigational correlation model data reported in ES.Scan.CorrParms*. All of these swath level covariance matrix related items [except GM_SigmaT, which is a fundamental sensor configuration driven statistical quantity constant for a swath] are to be understood as effective, post processing values that can be used in a formal error propagation methodology that is structurally the same as a classical pre-processing error propagation.



While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of X is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.X(j,1) = *a*, the large time asymptotic value

ES.Swath.CorrParms.X(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.X(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.X(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of

negligible numerical significance.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will be omitted.

Note that ES.Swath.CorrParms.X(k) et al should be[124] populated with values that are corrected for registration impacts due to processing. The burden of making useful data to represent the temporal correlation levels of blocks within a processed data set belongs to the algorithm that computes the registration adjusted correlation times. Realistically, the break is between "fully correlated for practical purposes" and "uncorrelated". The instantaneous, position pair decorrelation is always going to be a rough figure anyway. So, trying to handle the partially decorrelated separations will always be more safely handled by assuming complete decorrelation. Relying on ES.Swath.CorrParms.X et. al. enables the transition calculation for partial decorrelation, but it should never be fully trusted and the fully decorrelated case should be available as a reference for users in any case

If ES.Swath.CorrParms.* is not populated, then the data must treated as not available and should not be taken to be zero.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.X(k) ES.Swath.CorrParms.Alpha(j,k)

20.2.115 ES.Swath.CorrParms.Y(j,k) [L2,3,4,5]

Brief Definition ES.Scan.CorrParms.Y(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net sensor position y-coordinate in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally

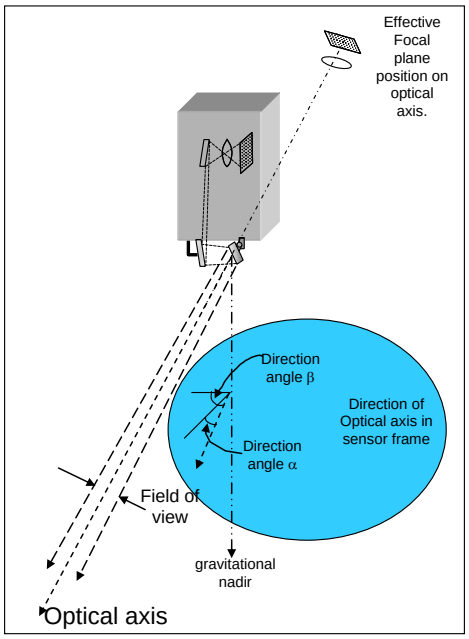
[124] If these corrections are not accounted for, the scan level and swath level time correlation data are the same and we could just as well use the ES.Scan. * data to represent the degree of correlation in geolocation error between two points (collected at different times) within the full dataset. In such a case, populating ES.Swath.CorrParms.X(k) with redundant data serves to support proper relative error measurement. We could get the same answers from the scan level data, but we would not know whether the processing did not change things or whether it could not. By populating ES.Swath.CorrParms* the scan and swath data always have the same meanings. If the swath data are not populated, it is because they could not be, not because the data are redundant. This avoids introducing a flag just to say that the scan level correlation and swath level correlations have the same values.

from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Swath.CorrParms.Y(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor y-position as determined at each moment of sampling the error covariance matrix. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation associated with there being improvements due to registration or other processing[123] that could make errors correlated over longer time intervals.
It may be that the registration processing filter uses a constant integration time and that all of the samples of ES.Swath.CorrParms.Y are the same. [In this case there will be only one set of ES.Swath.CorrParms.Y correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the processing uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

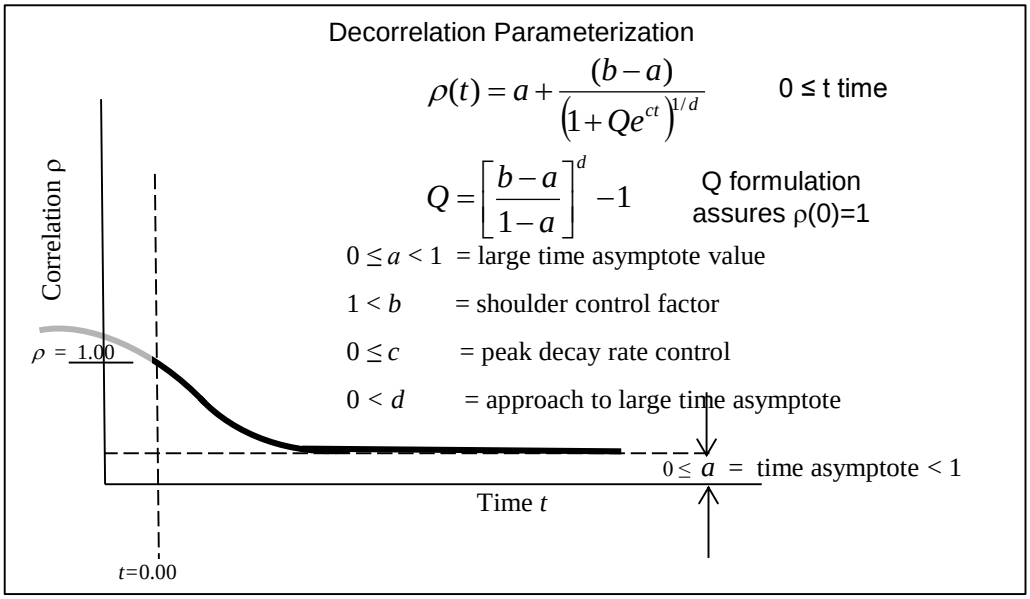
The quantities given by ES.Swath.CorrParms.Y describe the error correlation of a net sensor position measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.



While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of Y is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.Y(j,1) = *a*, the large time asymptotic value

ES.Swath.CorrParms.Y(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.Y(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.Y(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of

negligible numerical significance.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will be omitted.

Note that ES.Swath.CorrParms.Y(k) et. al. should be[125] populated with values that are corrected for registration impacts due to processing. The burden of making useful data to represent the temporal correlation levels of blocks within a processed data set belongs to the algorithm that computes the registration adjusted correlation times. Realistically, the break is between "fully correlated for practical purposes" and "uncorrelated". The instantaneous, position pair decorrelation is always going to be a rough figure anyway. So, trying to handle the partially decorrelated separations will always be more safely handled by assuming complete decorrelation. Relying on ES.Swath.CorrParms.Y et. al. enables the transition calculation for partial decorrelation, but it should never be fully trusted and the fully decorrelated case should be available as a reference for users in any case

If ES.Swath.CorrParms.* is not populated, then the data must treated as not available and should not be taken to be zero.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.Y(k) ES.Swath.CorrParms.Alpha(j,k)

20.2.116 ES.Swath.CorrParms.Z(j,k) [L2,3,4,5]

Brief Definition ES.Scan.CorrParms.Z(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the net sensor position z-coordinate in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally

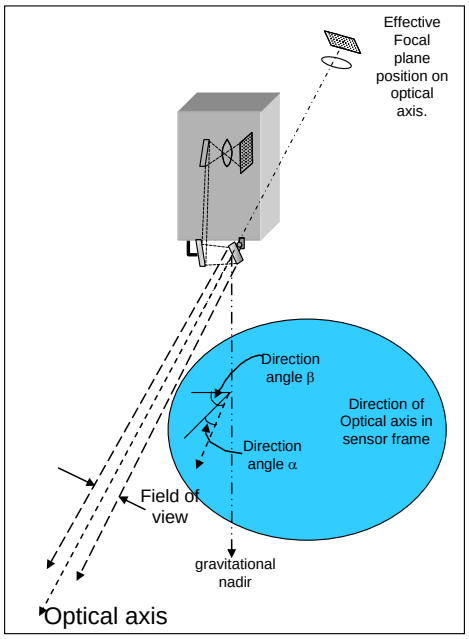
[125] If these corrections are not accounted for, the scan level and swath level time correlation data are the same and we could just as well use the ES.Scan. * data to represent the degree of correlation in geolocation error between two points (collected at different times) within the full dataset. In such a case, populating ES.Swath.CorrParms.X(k) with redundant data serves to support proper relative error measurement. We could get the same answers from the scan level data, but we would not know whether the processing did not change things or whether it could not. By populating ES.Swath.CorrParms* the scan and swath data always have the same meanings. If the swath data are not populated, it is because they could not be, not because the data are redundant. This avoids introducing a flag just to say that the scan level correlation and swath level correlations have the same values.

from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time.
ES.Swath.CorrParms.Z(j,k) are used to compute the level of correlation over time for the error in the earth frame sensor y-position as determined at each moment of sampling the error covariance matrix. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation associated with there being improvements due to registration or other processing[123] that could make errors correlated over longer time intervals.
It may be that the registration processing filter uses a constant integration time and that all of the samples of ES.Swath.CorrParms.Z are the same. [In this case there will be only one set of ES.Swath.CorrParms.Z correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the processing uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

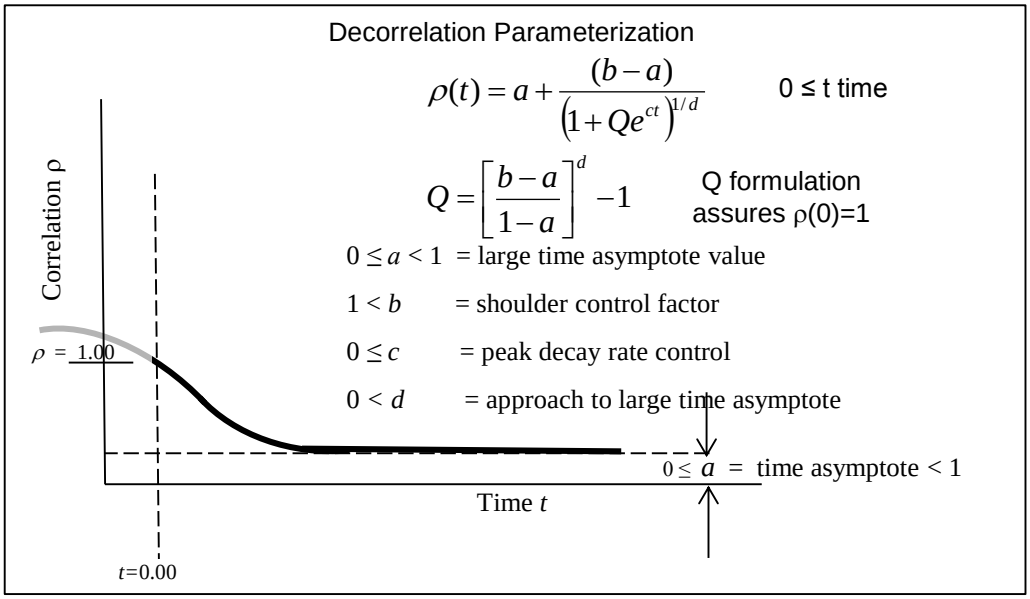
The quantities given by ES.Swath.CorrParms.Z describe the error correlation of a net sensor position measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.



While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of Z is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{\alpha t}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.Z(j,1) = *a*, the large time asymptotic value

ES.Swath.CorrParms.Z(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.Z(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.Z(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of

negligible numerical significance.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will be omitted.

Note that ES.Swath.CorrParms.Z(k) et. al. should be[125] populated with values that are corrected for registration impacts due to processing. The burden of making useful data to represent the temporal correlation levels of blocks within a processed data set belongs to the algorithm that computes the registration adjusted correlation times. Realistically, the break is between "fully correlated for practical purposes" and "uncorrelated". The instantaneous, position pair decorrelation is always going to be a rough figure anyway. So, trying to handle the partially decorrelated separations will always be more safely handled by assuming complete decorrelation. Relying on ES.Swath.CorrParms.Z et. al. enables the transition calculation for partial decorrelation, but it should never be fully trusted and the fully decorrelated case should be available as a reference for users in any case

If ES.Swath.CorrParms.* is not populated, then the data must treated as not available and should not be taken to be zero.

Governing quality metrics

See Also

PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.Z(j) registration impact ES.Swath.CorrParms.Alpha(j,k)

20.2.117 ES.Swath.CorrParms.R(j,n,k) [L2,3,4,5]

Brief Definition

ES.Swath.CorrParms.R(j,n,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error in the range measurement from the sensor to an output (likely synthetic) lidar point.

indices

Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.
Index, n, equals either [0,1] or [1,2] depending on the conventions of the implementing software code. The lower value of n carries data for ranging errors that do not grow with the range and the high value carries data that corresponds to an error that is proportional to the range itself.

Units/representation: Variable, single float

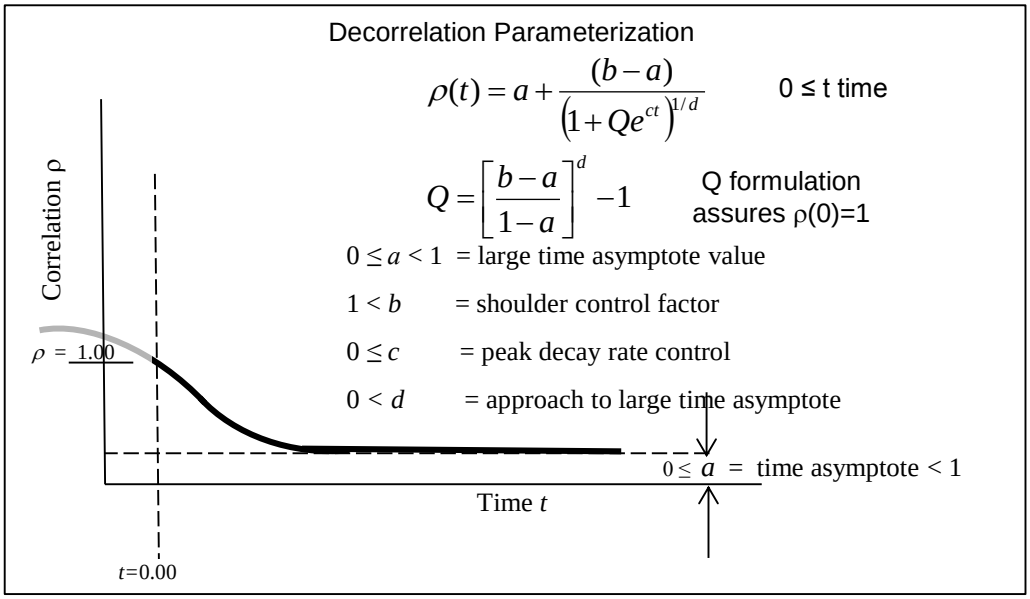
Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, not only is the uncertainty in the data coming out of the navigation and steering/attitude control system required, but the range uncertainty, itself, is needed. Furthermore, the error in relative distance between points measured at different times requires knowledge of the degree to which these errors are correlated over time. ES.Swath.CorrParms.R(j,n,k) are used to compute the level of correlation over time for the error in the earth frame sensor to lidar point range as determined at each moment of sampling the error covariance matrix. The 'ES.Swath' prefix indicates that this is data at the stage of sub-swath aggregation associated with there being improvements due to registration or other processing[123] that could make errors correlated over longer time intervals.
It may be that the atmospheric model is constant for a collect so that all of the (j indexed) samples of ES.Swath.CorrParms.R(j,n,k) are the same. [In this case there will be only one set of ES.Swath.CorrParms.R correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the lidar system tracks atmospheric model parameters and thus the ranging uncertainty could vary over a data collection.

The quantities given by ES.Scan.CorrParms.R describe the error correlation of a net ranging error resulting from applying the impact of refractive index estimation on 'rays' and round trip timing error contributions that are separated into path proportional and simply random error in estimating the ends of the roundtrip timing interval.

While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The auto correlation of R is modeled according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.R(j,1) = *a*, the large time asymptotic value

ES.Swath.CorrParms.R(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.R(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.R(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of

negligible numerical significance.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will be omitted.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Scan.CorrParms.R(j,n,k) registration impact ES.Swath.CorrParms.Alpha(j,k)

20.2.118 ES.Swath.CorrParms.AlphaBeta(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.AlphaBeta(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and beta in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

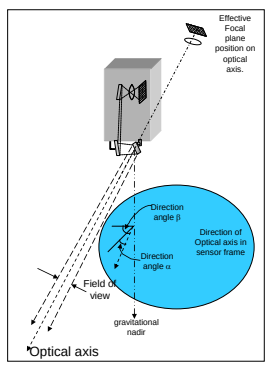
Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.AlphaBeta(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor orientation angles alpha and beta as determined at each moment of sampling the error covariance matrix.
The time dependence of cross correlation defined as

$$\rho_{\alpha\beta}(t = t_2 - t_1) = \left\langle \left(\alpha_{t_1} - \mu_{\alpha_{t_1}}\right)\left(\beta_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{a_{t_1}} \sigma_{\beta_{t_2}}\right)$$

The ‘ES.Swath’ prefix indicates that this is data at the stage of sub-swath aggregation associated with there being improvements due to registration or

other processing[123] that could make errors correlated over longer time intervals.
It may be that the registration processing filter uses a constant integration time and that all of the samples of ES.Swath.CorrParms.AlphaBeta are the same. [In this case there will be only one set of ES.Swath.CorrParms.AlphaBeta correlation model parameters (e.g., j=only 1), which should be used for the entire dataset.] The CMMD data design does permit the case where the processing uses an adaptive filter length and thus needs to be able to report multiple sets of correlation parameter values for a single collection operation.

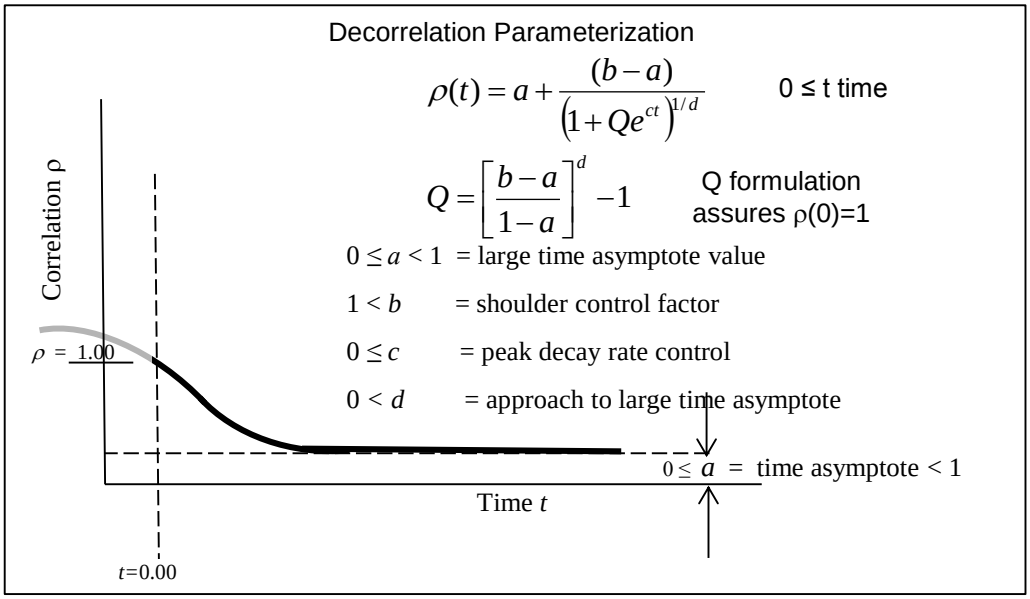
The quantities given by ES.Swath.CorrParms.AlphaBeta describe the error cross correlation of a net sensor orientation measure that includes the effect of the position of the sensor "box" itself (i.e. the platform position and orientation is included) and the steering system that directs the beam optical axis relative to the sensor.



While the system's post-processing correlation decay for any element of the covariance matrix could be known only numerically or as a design specific function, the correlation decay model parameterization presented here will be used to *fit* the class of suitable decorrelation behaviors to a simple functional representation that affords reasonably close (<10% peak) calculation of the correlations. The cross correlation of alpha and beta as a function of time between measurements within a single covariance snapshot is modeled, like the autocorrelations, according to

$$\rho(t) = a + \frac{b-a}{\left(1+Qe^{ct}\right)^{1/d}}$$

where the time t is in seconds and Q is constrained per the following diagram:



For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.AlphaBeta(j,1) = *a*, the large time asymptotic value

ES.Swath.CorrParms.AlphaBeta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.AlphaBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.AlphaBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

When the times of measurement of two points (t_1, t_2) for which we desire to know the error correlation both lie in the same snapshot interval, j, the evaluation of the correlation is straightforward and simple. The input parameter *t* of the correlation model is given by $t = (t_2 - t_1) \ge 0$. If the points in a pair have error correlation modeled by a single ES.Swath.CorrParms snapshot the asymptote may be used to evaluate fully decayed correlations.

In the case that we are looking at correlations in swath level processed data, and the times for a point pair belong to different post processing covariance snapshots, the correlation should be based on the asymptotic values of the correlation decay. Ideally, after internal registations, all the snapshots have the same asymptotic value and it should be near unity. Averaging the asymptotic values in this case should be considered both acceptable and of

negligible numerical significance.

If ES.Swath.CorrParms.AlphaBeta(j,k) is not populated, the cross correlation of alpha and beta is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.AlphaBeta(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.119 ES.Swath.CorrParms.AlphaGamma(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.AlphaGamma(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor orientations alpha and gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.AlphaGamma(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor orientation angles alpha and gamma as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{\alpha\gamma}(t = t_2 - t_1) = \left\langle \left(\alpha_{t_1} - \mu_{\alpha_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{a_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Swath.CorrParms.AlphaGamma(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Swath' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.AlphaGamma(j,1) = *a*, the large time asymptotic value
ES.Swath.CorrParms.AlphaGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.AlphaGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.AlphaGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.AlphaGamma(j,k) is not populated, the cross correlation of alpha and gamma is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also

PD.HiIndx.Swath ES.Swath.DelT_Cov
ES.Swath.CorrParms.AlphaGamma(j)
ES.Swath.CorrParms.Alpha(j,k)

20.2.120 ES.Swath.CorrParms.BetaGamma(j,k) [L2,3,4,5]

Brief Definition

ES.Swath.CorrParms.BetaGamma(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor orientations beta and gamma in the earth frame.

indices

Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.

ES.Swath.CorrParms.BetaGamma(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor orientation angles Beta and gamma as determined at each moment of sampling the error covariance matrix.

The time dependent cross correlation is defined as

$$\rho_{\beta\gamma}(t = t_2 - t_1) = \left\langle \left(\beta_{t_1} - \mu_{\beta_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{\beta_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Swath.CorrParms.BetaGamma(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.BetaGamma(j,1) = *a*, the large time asymptotic value.

ES.Swath.CorrParms.BetaGamma(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.BetaGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.BetaGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.BetaGamma(j,k) is not populated, the cross correlation of beta and gamma is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.BetaGamma(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.121 ES.Swath.CorrParms.XY(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.XY(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and y in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.XY(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coorldinates x and y as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{xy}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(y_{t_2} - \mu_{y_{t_2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{y_{t_2}}\right)$$

The description of ES.Swath.CorrParms.XY(j,k) follows in close analogy to

the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Swath' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.XY(j,1) = *a*, the large time asymptotic value.

ES.Swath.CorrParms.XY(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.XY(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.XY(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.XY(j,k) is not populated, the cross correlation of xand y is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.XY(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.122 ES.Swath.CorrParms.XZ(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.XZ(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinates x and z in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally

from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.XZ(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor coordinates x and z as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{xz}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(z_{t_2} - \mu z_{t_2}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{z_{t_2}}\right)$$

The description of ES.Swath.CorrParms.XZ(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.XZ(j,1) = *a*, the large time asymptotic value.
ES.Swath.CorrParms.XZ(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.XZ(j,3) = *c*, the peak decay control for the correlation

function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.XZ(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.XZ(j,k) is not populated, the cross correlation of x and z is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.XZ(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.123 ES.Swath.CorrParms.XAlpha(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.XAlpha(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation alpha in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.XAlpha(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor positioni coordinate x and angle alpha as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{x\alpha}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(\alpha_{t_2} - \mu_{\alpha_{t_2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{\alpha_{t_2}}\right)$$

The description of ES.Swath.CorrParms.XAlpha(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Swath' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.XAlpha(j,1) = *a*, the large time asymptotic value.

ES.Swath.CorrParms.XAlpha(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.XAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.XAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.XAlpha(j,k) is not populated, the cross correlation of x and alpha is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also

PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.XAlpha(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.124 ES.Swath.CorrParms.XBeta(j,k) [L2,3,4,5]

Brief Definition

ES.Swath.CorrParms.XBeta(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time

for the error cross covariance in the net sensor position coordinate x and orientation beta in the earth frame.

indices

Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation

Variable, single float

Treatment/usage

The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.

ES.Swath.CorrParms.XBeta(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate x and angle beta as determined at each moment of sampling the error covariance matrix.

The time dependent cross correlation is defined as

$$\rho_{x\beta}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(\beta_{t_2} - \mu_{\beta_{t2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{\beta_{t2}}\right)$$

The description of ES.Swath.CorrParms.XBeta(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.XBeta(j,1) = *a*, the large time asymptotic value.

ES.Swath.CorrParms.XBeta(j,2) = *b*, the “shoulder” control factor which

must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.XBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.XBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.XBeta(j,k) is not populated, the cross correlation of x and beta is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also: PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.XBeta(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.125 ES.Swath.CorrParms.XGamma(j,k) [L2,3,4,5]

Brief Definition: ES.Swath.CorrParms.XGamma(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate x and orientation gamma in the earth frame.

indices: Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.XGamma(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor

position coordinate x and angle gamma as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{x\gamma}(t = t_2 - t_1) = \left\langle \left(x_{t_1} - \mu_{x_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{x_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Swath.CorrParms.XGamma(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Swath' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.XGamma(j,1) = *a*, the large time asymptotic value.
ES.Swath.CorrParms.XGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.XGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.XGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.XGamma(j,k) is not populated, the cross correlation of x and gamma is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also

PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.XGamma(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.126 ES.Swath.CorrParms.YZ(j,k) [L2,3,4,5]

Brief Definition

ES.Swath.CorrParms.YZ(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinates y and z in the earth frame.

indices

Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation

Variable, single float

Treatment/usage

The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.

ES.Swath.CorrParms.YZ(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinates y and z as determined at each moment of sampling the error covariance matrix.

The time dependent cross correlation is defined as

$$\rho_{yz}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(z_{t_2} - \mu_{z_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{z_{t_2}}\right)$$

The description of ES.Swath.CorrParms.YZ(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be

populated as follows:
ES.Swath.CorrParms.YZ(j,1) = *a*, the large time asymptotic value
ES.Swath.CorrParms.YZ(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.YZ(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.YZ(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.YZ(j,k) is not populated, the cross correlation of y and z is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also: PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.YZ(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.127 ES.Swath.CorrParms.YAlpha(j,k) [L2,3,4,5]

Brief Definition: ES.Swath.CorrParms.YAlpha(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle alpha in the earth frame.

indices: Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the

product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.YAlpha(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate y and orientation angle alpha as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{y\alpha}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(\alpha_{t_2} - \mu_{\alpha_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{\alpha_{t_2}}\right)$$

The description of ES.Swath.CorrParms.YAlpha(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Swath' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.YAlpha(j,1) = *a*, the large time asymptotic value.
ES.Swath.CorrParms.YAlpha(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.YAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.YAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.YAlpha(j,k) is not populated, the cross correlation of y and alpha is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also

PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.YAlpha(j)

ES.Swath.CorrParms.Alpha(j,k)

20.2.128 ES.Swath.CorrParms.YBeta(j,k) [L2,3,4,5]

Brief Definition	ES.Swath.CorrParms.YBeta(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle beta in the earth frame.
indices	Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty. Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.
Units/representation	Variable, single float
Treatment/usage	The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame. ES.Swath.CorrParms.YBeta(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate y and orientation angle beta as determined at each moment of sampling the error covariance matrix. The time dependent cross correlation is defined as $\rho_{y\beta}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(\beta_{t_2} - \mu_{\beta_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{\beta_{t_2}}\right)$ The description of ES.Swath.CorrParms.YBeta(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of : • the ‘ES.Swath’ prefix • case that all the snapshots would have the same values of affected parameters • what physical effects are included in the calculation of this time dependent cross correlation • Significance of the parameters of the represented cross correlation • Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.YBeta(j,1) = *a*, the large time asymptotic value.

ES.Swath.CorrParms.YBeta(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.YBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.YBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.YBeta(j,k) is not populated, the cross correlation of y and beta is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.YBeta(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.129 ES.Swath.CorrParms.YGamma(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.YGamma(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate y and orientation angle gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.

Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the

absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.YGamma(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate y and orientation angle gamma as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{y\gamma}(t = t_2 - t_1) = \left\langle \left(y_{t_1} - \mu_{y_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{y_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Swath.CorrParms.YGamma(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.YGamma(j,1) = *a*, the large time asymptotic value.
ES.Swath.CorrParms.YGamma(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.YGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.YGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.YGamma(j,k) is not populated, the cross correlation of y and gamma is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There

the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.YGamma(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.130 ES.Swath.CorrParms.ZAlpha(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.ZAlpha(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle alpha in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.ZAlpha(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate z and orientation angle alpha as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{z\alpha}(t = t_2 - t_1) = \left\langle \left(z_{t_1} - \mu_{z_{t_1}}\right)\left(\alpha_{t_2} - \mu_{\alpha_{t_2}}\right)\right\rangle / \left(\sigma_{z_{t_1}} \sigma_{\alpha_{t_2}}\right)$$

The description of ES.Swath.CorrParms.ZAlpha(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time

dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.ZAlpha(j,1) = *a*, the large time asymptotic value
ES.Swath.CorrParms.ZAlpha(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.ZAlpha(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.ZAlpha(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive

If ES.Swath.CorrParms.ZAlpha(j,k) is not populated, the cross correlation of z and alpha is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also: PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.ZAlpha(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.131 ES.Swath.CorrParms.ZBeta(j,k) [L2,3,4,5]

Brief Definition: ES.Swath.CorrParms.ZBeta(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle beta in the earth frame.

indices: Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation: Variable, single float

Treatment/usage: The sensor system position and look direction data are used to convert range

and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.ZBeta(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate z and orientation angle beta as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{z\beta}(t = t_2 - t_1) = \left\langle \left(z_{t_1} - \mu_{z_{t_1}}\right)\left(\beta_{t_2} - \mu_{\beta_{t2}}\right)\right\rangle / \left(\sigma_{z_{t_1}} \sigma_{\beta_{t2}}\right)$$

The description of ES.Swath.CorrParms.ZBeta(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the ‘ES.Swath’ prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:
ES.Swath.CorrParms.ZBeta(j,1) = *a*, the large time asymptotic value.
ES.Swath.CorrParms.ZBeta(j,2) = *b*, the “shoulder” control factor which must be greater than unity. This determines the early correlation decay rate second derivative.
ES.Swath.CorrParms.ZBeta(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.
ES.Swath.CorrParms.ZBeta(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.ZBeta(j,k) is not populated, the cross correlation of z and beta is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.ZBeta(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.132 ES.Swath.CorrParms.ZGamma(j,k) [L2,3,4,5]

Brief Definition ES.Swath.CorrParms.ZGamma(j,k) gives the values of the parameters of the post-processing geopositioning error correlation model as a function of time for the error cross covariance in the net sensor position coordinate z and orientation angle gamma in the earth frame.

indices Index j, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty.
Index, k, ranges from PD.LoIndx.Swath(12) to PD.HiIndx.Swath(12), that is over the parameters of the swath level error correlation model.

Units/representation Variable, single float

Treatment/usage The sensor system position and look direction data are used to convert range and ray information to earth frame XYZ position. To support rigorous error propagation, the uncertainty in this data coming out of the navigation and steering/attitude control system is needed. Moreover, the correlations of the absolute sensor position errors are effectively altered when portions of the 3-D image or point cloud are registered in processing of the data. When registration reduces relative errors, this is equivalent to increasing the correlation of absolute errors while only reducing their magnitude when the product is more accurately positioned in the earth frame.
ES.Swath.CorrParms.ZGamma(j,k) are used to compute the post-processing level of cross correlation over time for the errors in the earth frame sensor position coordinate z and orientation angle gamma as determined at each moment of sampling the error covariance matrix.
The time dependent cross correlation is defined as

$$\rho_{z\gamma}(t = t_2 - t_1) = \left\langle \left(z_{t_1} - \mu_{z_{t_1}}\right)\left(\gamma_{t_2} - \mu_{\gamma_{t_2}}\right)\right\rangle / \left(\sigma_{z_{t_1}} \sigma_{\gamma_{t_2}}\right)$$

The description of ES.Swath.CorrParms.ZGamma(j,k) follows in close analogy to the description of ES.Swath.CorrParms.AlphaBeta(j,k). See that definition paragraph for the significance of :

- the 'ES.Swath' prefix
- case that all the snapshots would have the same values of affected parameters
- what physical effects are included in the calculation of this time dependent cross correlation
- Significance of the parameters of the represented cross correlation
- Computing time dependent cross correlation across covariance snapshot boundaries

For the case that PD.LoIndx.Swath(12)=1, the model parameters would be populated as follows:

ES.Swath.CorrParms.ZGamma(j,1) = *a*, the large time asymptotic value.

ES.Swath.CorrParms.ZGamma(j,2) = *b*, the "shoulder" control factor which must be greater than unity. This determines the early correlation decay rate second derivative.

ES.Swath.CorrParms.ZGamma(j,3) = *c*, the peak decay control for the correlation function, which must be positive {or zero if there is no decay of the correlation}.

ES.Swath.CorrParms.ZGamma(j,4) = *d*, the control factor for the approach to the large time asymptote, which must be positive.

If ES.Swath.CorrParms.ZGamma(j,k) is not populated, the cross correlation of z and gamma is to be treated as time independent.

All error covariance related data will be defined at the scan level too. There the impacts of processing and re-filtering the data will not be represented.

Governing quality metrics

See Also PD.HiIndx.Swath ES.Swath.DelT_Cov ES.Swath.CorrParms.ZGamma(j) ES.Swath.CorrParms.Alpha(j,k)

20.2.133 ES.Swath.DelT_Cov[L2,3,4,5]

Brief Definition ES.Swath.DelT_Cov is the time increment between post processing error covariance data "snapshots"

indices none

Units/representation Seconds, single float

Treatment/usage

For this lidar data definition scheme, the post-processing[126] error covariance data and decorrelation times derived by processing the 3-D data via registration techniques are reported on a uniform time grid. At each time hack or sampling time, the available sigmas, cross correlations and decorrelation times output as auxiliary information by the processor system's software constitute a snap shot of the revised uncertainty data. The error propagation model defines the transformations required to convert such data to error ellipsoids on absolute and relative geometric measurements.

The time interval between reported samples of this error covariance is given by ES.Swath.DelT_Cov. The value of ES.Swath.DelT_Cov is determined by the error propagation data support module in the product generating application so that meaningful variations in the covariance data during a data set collection interval can be represented. The total number of "snap shots", Ns, of covariance data defines (Ns-1) sub-intervals of the total data set collection time is derived from PD.HiIndx.Swath.
. The post-processing error covariance at any particular time is determined by a suitable interpolation of the sequence of error covariance snap shots available at the ES.Swath.DelT_Cov intervals. The product ES.Swath.DelT_Cov × (Ns-1) is equal to the total collection time for the data set. Thus there will be a sample of error covariance representing the start time of collection and another sample of covariance representing the end time of collection. The application that populates ES.Swath.DelT_Cov must make the interval small enough that linear interpolation of every error covariance matrix element gives a fair representation (within 10%) of the instantaneous value of error covariance.

Governing quality metrics

See Also

ES.Scan.Delt_Cov PD.HiIndx.Swath

20.2.134 ES.Swath.GM_SigmaRfrac(i,j) [L2,3,4,5]

Brief Definition

ES.Swath.GM_SigmaRfrac(i,j) provides the fraction of the raw point range uncertainty remaining after processing to reduce the impact of a range point spread due to sampling the range at intervals finer than transmit pulse length.

indices

Index, i, ranges from 1 to 10 (or 0 to 9 depending on the convention of the processor that is used to mitigate transmit time uncertainties for detected

[126] "post-processing" used in this context refers to values that apply when the process of creating a 3-D point cloud of measured positions has made use of adaptive corrections such as registrations or block adjustments that would tend to reduce geolocation errors from the native impacts of geometric projections necessary to "propagate" errors from sensor space to the product earth frame.

photons in Geiger mode)
Index , j, 1,2 (or 0 to 1 on same dependency as i)

Units/representation
Dimensionless, single float

Treatment/usage
ES.Swath.GM_SigmaT, defined below, determines the range over which points for a surface at a single range from the sensor are spread in a raw Geiger mode point cloud. After processing, a fraction of the spreading remains in all of the local range estimates within a small homogeneous region of the point cloud. ES.Swath.GM_SigmaRfrac is that residual spreading given as a fraction of the ES.Swath.GM_SigmaT left uncorrected. The local range uncertainty near a particular scene point is determined by multiplying the suitably selected value of ES.Scan.GM_SigmaR(j) by the suitably selected value of residual uncertainty fraction derived from ES.Swath.GM_SigmaRfrac(i,1) and ES.Swath.GM_SigmaRfrac(i,2), which are described in the following paragraph.

ES.Swath.GM_SigmaRfrac(i,j) is provided as a table of 10 rows and 2 columns. Each row corresponds to a 10% range of normalized intensity (aka return fraction) assigned to a processed point. For row, i, (we illustrate here with i=1 to 10), the normalized intensity (returns/interrogations with no path extinction correction in a voxel) will cover the range of (i-1)*10% to i*10%. So, i= 1 covers the normalized intensity from 0 to 10% and i=10 covers the normalized intensity from 90% to 100%.

ES.Swath.GM_SigmaRfrac(i,1) = the processed residual range uncertainty fraction based on a nominal (80% of scene) denormalization of the intensity. Denormalization is inversion from return ratio (returns/interrogations) to the point count used in removing the lidar's native range point spread (see SC.WF.TxProfile.Dat(il,j)). The populated value of residual range uncertainty fraction is averaged over the intensity range for the value of the "i" index.

ES.Swath.GM_SigmaRfrac(i,2) = the processed residual range uncertainty fraction based on relatively highly sampled regions (densest 20% of scene) denormalization of the intensity. Denormalization is inversion from return ratio (returns/interrogations) to the point count used in removing the lidar's native range point spread (see SC.WF.TxProfile.Dat(il,j)). The populated value of residual range uncertainty fraction is averaged over the intensity range for the value of the "i" index.

Governing quality metrics

See Also
ES.Swath.GM_SigmaT ES.Scan.GM_SigmaR(j) SC.WF.TxProfile.Dat(il,j)

20.2.135 ES.Swath.GM_SigmaT [L1,2,3,4,5]

Brief Definition	ES.Swath.GM_SigmaT provides the range independent round trip timing uncertainty information for a given Geiger Mode dataset.
indices	none
Units/representation	Seconds, single float
Treatment/usage	There are three sources of Geiger mode roundtrip timing error:

1. Error in the transmit time recorded
   Physical sources common to other modes are the lidar's sampling clock step size and delay between the actual pulse detection and recording/marking the event. A small contributor is clock jitter, which drives a zero mean variation in the clock step interval. A unique error source for GM is that we can't mark individual photons in an extended transmitted flash, so knowledge of the time of a received photon's transmission is uncertain. The distribution of this error can be quantified by the transmit pulse fast time profile, which would be populated in SC.WF.TxProfile.Dat(il,j).
2. Error in the receive time recorded
   Physical sources common to other modes are the lidar's sampling (duration of clock step/precision of recording receive time), detector response delay, incorrect calibration for receive delays due to signal propagating from focal plane detector to recorder interface. A small contributor is clock jitter, which drives a zero mean variation in the clock step interval.
3. Error in clock rate
   This represents an error in the mean clock step interval. This results in an accumulating bias to total measured roundtrip time intervals over short times (order of a lidar collection operation). The clock rate error itself is random over long periods of time (hours to days).

We collect the effects of these errors into two terms that can be combined to yield the uncertainty in roundtrip time at the swath level, that is over a single collection product. The first term, presented in ES.Swath.GM_SigmaT, covers the errors in identifying the end points of the roundtrip time interval. The second term accounts for the uncertainty in the clock rate, which leads to a contribution to roundtrip error that grows with the roundtrip time itself. That range dependent round trip time error is driven by the uncertainty supplied in SC.Swath.SigmaClkStep.

ES.Swath.GM_SigmaT is the time interval duration representing the standard deviation of the measured return time for photons returning from a single transmitted pulse and arriving from a reflecting surface at a single range.[127] It is assumed that a large enough number of photon receptions is used to determine this statistic that the standard deviation is a stable value as more measurements are added to the ensemble.

For Geiger mode operation, the uncertainty in roundtrip timing is due only to the sensor's ability to precisely and accurately record the transmission and return times of a photon. This can be primarily driven by the length of sampling clock intervals for recording events, but also has contributions from the rise time triggering process for a measured (Tx and Rx) pulse and pulse envelope. Because Geiger mode is not measuring photo current, there is no scatterer or scene structural component to the decision of when a return pulse has arrived to make the roundtrip time uncertainty pulse or return dependent. Thus, for a single data collect there is only one value of the roundtrip timing uncertainty, which is why this is treated as a swath level parameter. Unfortunately, it could be due to the sampling clock interval, in which case that value is to be derived from SC.Swath.RateFactor(ia) and SC.Swath.TimeStep, or it could be dominated by statistical fluctuations when the time step is small enough to resolve different flux/sensitivity[128] dependent trigger response delays. One phenomenon that feeds round trip timing error is assigning a *transmit* time to a pulse that does not have a square leading edge. If a photon can be transmitted in a clock interval that precedes the clock interval that gets assigned as the transmit time, its return from a fixed range target can be recorded by a Geiger mode sensor as a time that is too early for the actual range to the target. While we provide the transmit pulse power profile in SC.WF.TxProfile.Dat(il,j) to support processing mitigation of the roundtrip timing uncertainty, the roundtrip time uncertainty for a raw Geiger mode return needs to account for photons transmitted in advance of the recorded pulse transmission time. ES.Swath.GM_SigmaT is where this effect is incorporated, assuming that it is a significant player relative to other error sources for the lidar that collected the data in the dataset.
We don't know for sure the total flux incident for any particular pulse, so the statistical biases in the detector response time, while actually deterministic in

[127] "Single range" means that the scattering process off the target may not spread the return pulse by more than one tenth of the smallest (fast) time increment measurable by the lidar.

[128] e.g., the APD bias level and detector temperature. Any beam attenuation applied by the sensor, transmit flux variation, or scene reflectivity variations will affect the return flux density and thus the GM possibility of high return flux leading to a lower derived roundtrip time of the return. These effects are incorporated in ES.Swath.GM_SigmaT.

and of themselves, are often experienced in a random manner and may thus be considered to be contributors to a timing uncertainty. If the difference between low and high flux trigger times is small compared to the fast clock step, photons incident near the end of a clock interval could actually be counted in the following clock interval. If the clock steps are small compared to the variation in trigger times on a returned pulse, then we could be resolving these shifts temporally and the variation in return time measurements is dominated by the flux/sensitivity level and not the clock step. ES.Swath.GM_SigmaT gives the flux/sensitivity induced timing uncertainty. The user (or user tools) must then compare the sampling clock step and ES.Swath.GM_SigmaT to determine the timing uncertainty for the dataset. The lidar vendor/operator must incorporate all of these considerations to set the value of ES.Swath.GM_SigmaT.

Governing quality metrics

See Also ES.Point.LM_SigmaT(ipt) ES.Scan.GM_SigmaR(j) SC.Swath.RateFactor(ia) SC.Swath.TimeStep SC.WF.TxProfile.Dat(il,j) SC.Swath.SigmaClkStep

20.2.136 ES.Swath.RhoAlphaGamma(k) [L2,3,4,5]

Brief Definition ES.Swath.RhoAlphaGamma(k) is the post processing correlation between the uncertainties in cos(alpha) and the angle gamma, and is part of the sensor error covariance data.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single float

Treatment/usage ES.Swath.RhoAlphaGamma(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoAlphaGamma(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation in the uncertainty in cosine(alpha) and the uncertainty in the angle, gamma, for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.

In this term, the net effect of the combined platform and sensor state on the

line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to any adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]
The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame) to the line of sight.
The angle gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also: ES.Swath.DelT_Cov ES.Scan.RhoAlphaGamma(j)

20.2.137 ES.Swath.RhoXBeta(k) [L2,3,4,5]

Brief Definition: ES.Swath.RhoXBeta(k) is the post processing correlation between the uncertainties in the x-position coordinate and the cosine of the angle beta, and is part of the sensor error covariance data.

indices: Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation: Dimensionless, single float

Treatment/usage: ES.Swath.RhoXBeta(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Swath.RhoXBeta(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation between the uncertainties in the sensor x-position and cosine(beta) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the covariance matrix

are less than 10%[129] when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]
The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) to the line of sight.
The x- position subject of this statistic is measured in the positive (increases to north for NED frame) horizontal x-direction in the earth frame at the sensor position at the time associated with index, k, value of the error covariance sample.

Governing quality metrics

See Also: ES.Swath.DelT_Cov ES.Scan.RhoXBeta(j)

20.2.138 ES.Swath.RhoAlphaBeta(k) [L2,3,4,5]

Brief Definition: ES.Swath.RhoAlphaBeta(k) is the post-processing correlation between the uncertainties in the cosines of the line of sight angles, alpha and beta, and is part of the sensor error covariance data.

indices: Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation: Dimensionless, single float

Treatment/usage: ES.Swath.RhoAlphaBeta(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and

[129] If this ever proves to a be a capability limiting convention, we could add a new metadata item that is the interpolation accuracy provided and have this paragraph point to that new parameter. Need is not sufficiently likely to add the item now.

range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoAlphaBeta(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation between the uncertainties in cosine(alpha) and cosine(beta) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position is represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments, such as by registration, not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The angle, alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame) to the line of sight.

The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) to the line of sight in the earth frame at the sensor position at the time associated with the index, k, value of the covariance sample.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.RhoAlphaBeta(j)

20.2.139 ES.Swath.RhoBetaGamma(k) [L2,3,4,5]

Brief Definition ES.Swath.RhoBetaGamma(k) is the post-processing correlation between the uncertainties in cos(beta) and the angle,gamma, and is part of the sensor error covariance data.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-

processing position and LOS uncertainty

Units/representation: Dimensionless, single float

Treatment/usage: ES.Swath.RhoBetaGamma(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoBetaGamma(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation between the uncertainties in cosine(alpha) and the angle, gamma, for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. north for a NED frame)to the line of sight.

The angle gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also: ES.Swath.DelT_Cov ES.Scan.RhoBetaGamma(j)

20.2.140 ES.Swath.RhoXAlpha(k) [L2,3,4,5]

Brief Definition: ES.Swath.RhoXAlpha(k) is the post-processing correlation between the uncertainties in the x-position coordinate and cosine of angle alpha, and is part of the sensor covariance data.

indices	Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Units/representation	Dimensionless, single float
Treatment/usage	ES.Swath.RhoXAlpha(k) is an off diagonal element of a matrix representing the error covariance of the sensior position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoXAlpha(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation between the uncertainties in the x-position and cosine(alpha) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values. In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the post processing covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.] The angle, alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame)to the line of sight. The x- position subject of this statistic is measured in the positive (increases to north for NED frame) horizontal x-direction in the earth frame at the sensor position at the time associated with index, k, value of the error covariance sample.
Governing quality metrics	
See Also	ES.Swath.DelT_Cov ES.Scan.RhoXAlpha(j)

20.2.141 ES.Swath.RhoXGamma(k) [L2,3,4,5]

Brief Definition	ES.Swath.RhoXGamma(k) is the post-processing correlation between uncertainties in the sensor x-position coordinate and the angle, gamma, and is part of the sensor error covariance data.
indices	Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Units/representation	Dimensionless, single float
Treatment/usage	ES.Swath.RhoXGamma(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoXGamma(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation between uncertainties in the x-position and angle, gamma, for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.] The sensor position x coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame) at the sensor . The angle, gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.
Governing quality metrics	
See Also	ES.Swath.DelT_Cov ES.Scan.RhoXGamma(j)

20.2.142 ES.Swath.RhoXY(k) [L2,3,4,5]

Brief Definition	ES.Swath.RhoXY(k) is the post-processing correlation between uncertainties in the sensor x-position coordinate and y-position coordinate, and is part of the sensor error covariance data.
indices	Index k ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Units/representation	Dimensionless, single float
Treatment/usage	ES.Swath.RhoXY(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoXY(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation between the uncertainties in the x-position and y-position for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.] The x-coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame) at the sensor. The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame) at the sensor.
Governing quality metrics	
See Also	ES.Swath.DelT_Cov ES.Scan.RhoXY

20.2.143 ES.Swath.RhoXZ(k) [L2,3,4,5]

Brief Definition	ES.Swath.RhoXZ(k) is the post-processing correlation between uncertainties in the sensor x-position coordinate and z-position coordinate, and is part of the sensor error covariance data.
indices	Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Units/representation	Dimensionless, single float
Treatment/usage	ES.Swath.RhoXZ(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoXZ(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation between the uncertainties in the x-position and z-position for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.] The x-coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame) at the sensor . The z-coordinate is measured along the vertical reference direction (i.e. down for a NED frame) at the sensor.
Governing quality metrics	
See Also	ES.Swath.DelT_Cov ES.Scan.RhoXZ(j)

20.2.144 ES.Swath.RhoYAlpha(k) [L2,3,4,5]

Brief Definition	ES.Swath.RhoYAlpha(k) is the post processing correlation between uncertainties in the y-position coordinate and cosine of angle alpha, and is part of the sensor error covariance data.
indices	Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Units/representation	Dimensionless, single float
Treatment/usage	ES.Swath.RhoYAlpha(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoYAlpha(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation between the uncertainties in the y-position and cosine(alpha) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values. In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.] The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame) to the line of sight at the sensor. The y- position subject of this statistic is measured in the positive (increases to east for NED frame) horizontal y-direction in the earth frame at the sensor position at the time associated with index, k, value of the error covariance sample.
Governing quality	

metrics
See Also ES.Swath.DelT_Cov ES.Scan.RhoYAlpha(j)

20.2.145 ES.Swath.RhoYBeta(k) [L2,3,4,5]

Brief Definition ES.Swath.RhoYBeta(k) is the post processsing correlation between uncertainties in the y-position coordinate and cosine of angle beta, and is part of the sensor error covariance data.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single float

Treatment/usage ES.Swath.RhoYBeta(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.

ES.Swath.RhoYBeta(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of correlation between the uncertainties in the y-position and cosine(beta) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the swath level error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction error covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) at the sensor to the line of sight.

The y- position subject of this statistic is measured in the positive (increases

	to east for NED frame) horizontal y-direction in the earth frame at the sensor position at the time associated with index, k, value of the error covariance sample.
Governing quality metrics	
See Also	ES.Swath.DelT_Cov ES.Scan.RhoYBeta(j)

20.2.146 ES.Swath.RhoYGamma(k) [L2,3,4,5]

Brief Definition	ES.Swath.RhoYGamma(k) is the post processing correlation between uncertainties in the sensor y-position coordinate and the angle, gamma, and is part of the sensor error covariance data.
indices	Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty
Units/representation	Dimensionless, single float
Treatment/usage	ES.Swath.RhoYGamma(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. ES.Swath.RhoYGamma(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments. The reported values of correlation between the uncertainties in the y-position and gamma for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values. In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction error covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame) at the sensor position.
The angle gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.RhoYGamma(j)

20.2.147 ES.Swath.RhoYZ(k) [L2,3,4,5]

Brief Definition ES.Swath.RhoYZ(k) is the post processing correlation between uncertainties in the sensor y-position coordinate and z-position coordinate, and is part of the sensor error covariance data.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single float

Treatment/usage ES.Swath.RhoYZ(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Swath.RhoYZ(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation between the uncertainties in y-position and z-position for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame) at the sensor .
The z-coordinate is measured along the vertical reference direction (i.e. down for a NED frame) at the sensor .

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.RhoYZ(j)

20.2.148 ES.Swath.RhoZAlpha(k) [L2,3,4,5]

Brief Definition ES.Swath.RhoZAlpha(k) is the post-processing correlation between uncertainties in the sensor z-position coordinate and the cosine of the angle alpha, and is part of the sensor error covariance data.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single float

Treatment/usage ES.Swath.RhoZAlpha(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Swath.RhoZAlpha(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation between the uncertainties in the z-position and cosine(alpha) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position is represented so that the post-processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Swath.xxx error covariance

data is after the introduction of processing generated corrections.]
The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame) to the line of sight at the sensor.
The z- position subject of this statistic is measured in the positive (increases downward for NED frame) vertical direction in the earth frame at the sensor position at the time associated with index, k, value of the error covariance sample.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.RhoZAlpha(j)

20.2.149 ES.Swath.RhoZBeta(k) [L2,3,4,5]

Brief Definition ES.Swath.RhoZBeta(k) is the post-processing correlation between uncertainties in the z-position coordinate and cosine of angle beta, and is part of the sensor error covariance data.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single float

Treatment/usage ES.Swath.RhoZBeta(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Swath.RhoZBeta(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation between uncertainties in the z-position and cosine(beta) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the j index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and horizontal position are represented so that the post-processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the “raw values” after some processing has occurred. When this is done, the error covariance data must

be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]
The angle, beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) to the line of sight.
The z- position subject of this statistic is measured in the positive (increases downward for NED frame) vertical direction in the earth frame at the sensor position at the time associated with index, k, value of the error covariance sample.

Governing quality metrics

See Also: ES.Swath.DelT_Cov ES.Scan.RhoZBeta(j)

20.2.150 ES.Swath.RhoZGamma(k) [L2,3,4,5]

Brief Definition: ES.Swath.RhoZGamma(k) is the post-processing correlation between uncertainties in the sensor z-position coordinate and angle, gamma, and is part of the sensor error covariance data.

indices: Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation: Dimensionless, single float

Treatment/usage: ES.Swath.RhoZGamma(k) is an off diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances.
ES.Swath.RhoZGamma(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of correlation between uncertainties in the z-position and gamma for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post- processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent

with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]
The z-coordinate is measured along the vertical reference direction (i.e. down for a NED frame) at the sensor.
The angle, gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also: ES.Swath.DelT_Cov ES.Scan.RhoZGamma(j)

20.2.151 ES.Swath.SigmaAlpha(k) [L2,3,4,5]

Brief Definition: ES.Swath.SigmaAlpha(k) is the post-processing standard deviation of the measurement of cosine of angle alpha, a contributor to the diagonal of the error covariance matrix for the sensor.

indices: Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation: Dimensionless, single float

Treatment/usage: ES.Swath.SigmaAlpha(k) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.
ES.Swath.SigmaAlpha(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the standard deviation of the measurement of cosine(alpha) for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post-processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx erroro covariance data is after the introduction of processing generated corrections.]

The angle alpha, whose cosine is the subject of this statistic, is measured from the horizontal x-reference direction (i.e. north for a NED frame) at the sensor to the line of sight.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.SigmaAlpha(j)

20.2.152 ES.Swath.SigmaBeta(k) [L2,3,4,5]

Brief Definition ES.Swath.SigmaBeta(k) is the post-processing standard deviation of the measurement of cosine of angle beta, a contributor to the diagonal of the error covariance matrix for the sensor.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single float

Treatment/usage ES.Swath.SigmaBeta(k) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.

ES.Swath.SigmaBeta(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of the standard deviation of the meausrement (i.e. uncertainty) of cosine(beta) for each value of index k, are successive time

samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post-processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The angle beta, whose cosine is the subject of this statistic, is measured from the horizontal y-reference direction (i.e. east for a NED frame) at the sensor to the line of sight.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.SigmaBeta(j)

20.2.153 ES.Swath.SigmaGamma(k) [L2,3,4,5]

Brief Definition ES.Swath.SigmaGamma(k) is the post-processing standard deviation of the measurement of angle gamma, a contributor to the diagonal of the error covariance matrix for the sensor.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation Dimensionless, single loat

Treatment/usage ES.Swath.SigmaGamma(k) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.
ES.Swath.SigmaGamma(k) represents a value that includes improvements

due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the standard deviation of the measurement of gamma for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post-processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The angle gamma, which is the subject of this statistic, is measured around the line of sight looking towards the sensor from the sensed position.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.SigmaGamma(j)

20.2.154 ES.Swath.SigmaX(k) [L2,3,4,5]

Brief Definition ES.Swath.SigmaX(k) is the post-processing standard deviation of the measurement of the x-coordinate of sensor position, a contributor to the diagonal of the error covariance matrix for the sensor.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation meters, single float

Treatment/usage ES.Swath.SigmaX(k) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error ovariance

matrix.
ES.Swath.SigmaX(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the uncertainty (standard deviation of the measurement) of X for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post- processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The sensor x-coordinate is measured along the horizontal x-reference direction (i.e. north for a NED frame) at the sensor.

Governing quality metrics

See Also: ES.Swath.DelT_Cov ES.Scan.SigmaX(j)

20.2.155 ES.Swath.SigmaY(k) [L2,3,4,5]

Brief Definition: ES.Swath.SigmaY(k) is the post-processing standard deviation of the measurement of the y-coordinate of sensor position, a contributor to the diagonal of the error covariance matrix for the sensor.

indices: Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation: meters, single float

Treatment/usage: ES.Swath.SigmaY(k) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be

multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.
ES.Swath.SigmaY(k represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.
The reported values of the uncertainty (standard deviation of the measurement) of Y for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.
In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the post-processing error covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post- correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The y-coordinate is measured along the horizontal y-reference direction (i.e. east for a NED frame) at the sensor.

Governing quality metrics

See Also ES.Swath.DelT_Cov ES.Scan.SigmaY(j)

20.2.156 ES.Swath.SigmaZ(k) [L2,3,4,5]

Brief Definition ES.Swath.SigmaZ(k) is the post-processing standard deviation of the measurement of the z-coordinate of sensor position, a contributor to the diagonal of the error covariance matrix for the sensor.

indices Index k, ranges from PD.LoIndx.Swath(1) to PD.HiIndx.Swath(1), generally from 1 to the number of snapshots of the error covariance matrix for post-processing position and LOS uncertainty

Units/representation meters, single float

Treatment/usage

ES.Swath.SigmaZ(k) is the square root of a diagonal element of a matrix representing the error covariance of the sensor position, line of sight and range data. The reported off diagonals (correlation coefficients) need to be multiplied by the appropriate standard deviations to obtain true covariances. Thus, we populate with the standard deviation rather than the variance. This item will need to be squared to obtain a diagonal entry of the error covariance matrix.

ES.Swath.SigmaZ(k) represents a value that includes improvements due to processing that might improve geometric fidelity, such as internal registrations or block adjustments.

The reported values of the uncertainty (standard deviation of the measurement) of Z for each value of index k, are successive time samples at intervals given by ES.Swath.DelT_Cov. This sampling must be populated sufficiently densely to assure that errors in the error covariance matrix are less than 10% when linearly interpolated to times different than that of the k index values.

In this term, the net effect of the combined platform and sensor state on the line of sight and rotation about that line of sight are represented so that the swath level covariance matrix is directly applicable to lidar points that have been subject to adjustments such as by registration not consistent with the reported position and line of sight data. [It is permitted that position and line of sight data might be updated from the "raw values" after some processing has occurred. When this is done, the error covariance data must be updated to be consistent. However, when the points have been shifted due to adaptive corrections and the sensor position and line of sight have not, we have the case that we wish access to both the pre-correction covariance and the post-correction covariance. In this latter situation, ES.Swath.xxx error covariance data is after the introduction of processing generated corrections.]

The z-coordinate is measured along the vertical reference direction (i.e. down for a NED frame) at the sensor.

Governing quality metrics

See Also

ES.Swath.DelT_Cov ES.Scan.SigmaZ(j)

20.2.157 ES.Swath.SensorBasedCoordFlg(j) [L2,3,4,5]

Brief Definition

ES.Swath.SensorBasedCoordFlg(j) indicates the coordinate system and frame orientation for point positions presented in the sensor frame.

indices

Index j=1,2

Units/representation Dimensionless, UInt8

Treatment/usage We separately define the coordinate system for points in the point cloud [ES.Swath.SensorBasedCoordFlg(1)], and the orientation of the orthogonal Cartesian frame axes [ES.Swath.SensorBasedCoordFlg(2)] for the case of sensor frame points.

The orthogonal, right handed, Cartesian frame axes are designated **x**,**y**,**z,** with the usual **x × y=z.**

To facilate the use of direction angles, we define the positive half space to be that part of the space where point positions obey $x \times y \geq 0$. Ranges, R, will be given a positive sign for points in the positive half space and a negative sign for points in the negative half space. $\|R\|$ is the distance from the origin (sensor reference position) to a point in the point cloud.

ES.Swath.SensorBasedCoordFlg(1) indicates the coordinate system choice:

Code	Meaning
0	0 (default[130]) means that coordinates are actually earthframe; the sensor position motion is already "built in" to create a stationary point cloud origin independent of the ECEF sensor position for each tile. The coordinates are understood to be XYZ (or, eventually quad+Z)
1	means that the j=1,2,3 {or 2,3,4 when the repeat flag is used} coordinate values in LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) (or) LD.Point.Pos.*TileNum*(ipt,j) correspond to (cosα,cosβ,R). The direction angle, α is measured from the frame base plane x axis. The direction angle, β is measured from the frame base plane y axis. The range, R, is positive for $z \geq 0$, even though the coordinates are not expresse using a z in this option.
2	means that the j=1,2,3 {or 2,3,4 when the repeat flag is used} coordinate values in LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) (or) LD.Point.Pos.*TileNum*(ipt,j) correspond to (α,β,R). The direction angle, α is measured from the frame base plane x axis. The direction angle, β is measured from the frame base plane y axis. The range, R, is positive for $z \geq 0$, even though the coordinates are not expressed using a z in this option

[130] This means that we may either leave ES.Swath.SensorBasedCoordFlg(j) unpopulated, or explicitly set ES.Swath.SensorBasedCoordFlg(1) to 0 to indicate that the point cloud is fixed in the earth frame.

3	means that the j=1,2,3 {or 2,3,4 when the repeat flag is used} coordinate values in LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) (or) LD.Point.Pos.*TileNum*(ipt,j) correspond to (xyz) i.e. sensor frame Cartesian coordinates.
4	means that the j=1,2,3 {or 2,3,4 when the repeat flag is used} coordinate values in LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) (or) LD.Point.Pos.*TileNum*(ipt,j) correspond to (ϕ,ρ,z) that is, cylindrical coordinates [ϕ is the azimuth angle in the base plane and is measured from the frame x axis. The range, ρ, is measured from the from the z axis, and z (height above the base plane) is measured along the z axis.
5	means that the j=1,2,3 {or 2,3,4 when the repeat flag is used} coordinate values in LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) (or) LD.Point.Pos.*TileNum*(ipt,j) correspond to spherical coordinates (θ,ϕ,$\Re$). This range, $\Re$, is the always positive distance from the origin. The cone angle, θ, is measured from the third (z) axis, and azimuth angle, ϕ, is measured from the x axis.

ES.Swath.SensorBasedCoordFlg(2) indicates the coordinate frame orientation. The **xyz** directions are determined by a choice of rules. Every tile follows the same rule indicated by the code value chosen from these options :

Code	**Meaning (rule)**
1	is the code for local ENU, (**x** axis= E, **y** axis=N, **z** axis=U)
2	is the code for orienting per the sensor "facing" direction. {we exploit the fact that the line of sight (that is, the direction the sensor is facing) is determined by the unit vectors SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j)=**x** and SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j)=**y**}. The sensor "faces" in the z direction
3	The **z** axis is oriented along the ECEF sensor velocity vector, with z understood to be in a nearly horizontal direction with respect to gravity in this case. The x axis is **-z** × **UP.** The **y** axis is **z**×**x**. **UP** is the local gravitational vertical.

4	The **x** axis is oriented along the ECEF velocity vector. The **y** axis is **-x** × **UP**. The z axis is x × y. **UP** is the local gravitational vertical.

ES.Swath.SensorBasedCoordFlg(2) should not be consulted for use unless ES.Swath.SensorBasedCoordFlg(1) is non-zero, indicating a need to know how the sensor frame coordinate axis is oriented.

Governing quality metrics

See Also PF.Point.SenFrmRefPulse(j_tile) LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) LD.Point.Pos.*TileNum*(ipt,j) ES.Align.XYZ_Flag

20.2.158 ES.Swath.HiConfdncP_Fac(i,j,k) [L1,2,3,4,5]

Brief Definition ES.Swath.HiConfdncP_Fac(i,j,k) multiplied by a standard deviation uncertainty (such as ES.Swath.SigmaT) provides the high confidence error bars of the quantity whose uncertainty is being provided.

indices Index, i, =1,2 {or 0,1} to cover two separate high confidence levels
Index, j runs from PD.LoIndx.Swath(13) to PD.HiIndx.Swath(13), that is over the number of conversion factor sets available for converting standard deviations to high confidence percentile values.
Index, k = 1,2 {or 0,1 per the coding conventions of the data data set writing application} cover the lower and upper error bars.

Units/representation dimensionless, single float

Treatment/usage Because the formalism for combining and propagating errors requires standard deviations[131] we provide basic uncertainties in the CMMD as standard deviations. But since capturing the significance of errors and regulating them to system levels that support reliable confidence in the data often needs a high percentile of conformance, we provide factors to convert (by multiplication) the fundamental mathematical uncertainty measure to the high confidence levels, such as the .90P or.95P level. Populating ES.Swath.HiConfdncP_Fac(i,j,k) makes it possible to display and use high confidence level error bars in addition to performing formal error propagation. The values of the high confidence percentiles carried in ES.Swath.HiConfdncP_Fac(i,j,k) are provided in PD.DatDscrptors.Uncerty(j).
To accommodate error distributions that are non-symmetric, we will need a factor for an upper error bar and a separate factor for a lower error bar.

[131] The square root of variance about the mean is a fundamental uncertainty measure. The algebra of error combination requires adding variances together,

These two values (indexed by k) form a "conversion factor set". For the case of a symmetrical error distribution, the two factors in a set will be equal. In general, the two error bars will be chosen to bracket the mean value of the measurand whose error is being described with the shortest range of errors. This assures that the most likely errors about the expected value are included in the high confidence ranges. For the case that the errors are normally distributed, ES.Swath.HiConfdncP_Fac(i,j,k,) values corresponding to PD.DatDscrptors.Uncerty(j) indicating .95P will be 1.959964 for both high and low error bars.
The ratio of, say, the .95P value to the standard deviation for range independent round trip timing errors is provided for the pre-processed data but is given an ES.Swath prefix because this is a single value for a collection operation and thus belongs to the swath.[132]

For the case of indexing starting from 1 for i, j and k:

<u>ES.Swath.HiConfdncP_Fac(1,1,1)</u>
Converts the standard deviation uncertainty of range independent round trip timing error in ES.Swath.SigmaT to the lower error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.
<u>ES.Swath.HiConfdncP_Fac(1,1,2)</u>
Converts the standard deviation uncertainty of range independent round trip timing error in ES.Swath.SigmaT to the upper error bar of a PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.
<u>ES.Swath.HiConfdncP_Fac(2,1,1)</u>
Converts the standard deviation uncertainty of range independent round trip timing error in ES.Swath.SigmaT to the lower error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.
<u>ES.Swath.HiConfdncP_Fac(2,1,2)</u>
Converts the standard deviation uncertainty of range independent round trip timing error in ES.Swath.SigmaT to the upper error bar of a PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.

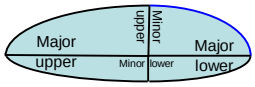
<u>ES.Swath.HiConfdncP_Fac(1,2,1):</u>
Converts the standard deviation uncertainty of clock rate error in SC.Swath.SigmaClkStep to the lower error bar of the

[132] This statement applies to single collection based products such as will be found at Enterprise level 2 or 3. For Enterprise level-4 multiple collection based products, there may be variations from tile to tile, depending on how the sources are sorted within the dataset.

PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean. Note that when the fast time sampling PD.DatDscrptors.Uncerty(1) clock error is desired, the SC.Swath.SigmaClkStep must be scaled by SC.Swath.RateFactor as well as ES.Swath.HiConfdncP_Fac(1,2,1).
ES.Swath.HiConfdncP_Fac(1,2,2):
Converts the standard deviation uncertainty of clock rate error in SC.Swath.SigmaClkStep to the upper error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean. Note that when the fast time sampling PD.DatDscrptors.Uncerty(1) clock error is desired, the SC.Swath.SigmaClkStep must be scaled by SC.Swath.RateFactor as well as ES.Swath.HiConfdncP_Fac(1,2,2).
ES.Swath.HiConfdncP_Fac(2,2,1):
Converts the standard deviation uncertainty of clock rate error in SC.Swath.SigmaClkStep to the lower error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean. Note that when the fast time sampling PD.DatDscrptors.Uncerty(2) clock error is desired, the SC.Swath.SigmaClkStep must be scaled by SC.Swath.RateFactor as well as ES.Swath.HiConfdncP_Fac(2,2,1).
ES.Swath.HiConfdncP_Fac(2,2,2):
Converts the standard deviation uncertainty of clock rate error in SC.Swath.SigmaClkStep to the upper error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean. Note that when the fast time sampling PD.DatDscrptors.Uncerty(2) clock error is desired, the SC.Swath.SigmaClkStep must be scaled by SC.Swath.RateFactor as well as ES.Swath.HiConfdncP_Fac(2,2,2).

The following data is provided to convert pre-computed ground covariance data based on standard deviations to high confidence error (quasi)[133] ellipsoid data. Note that there are (PD.HiIndx.Point.SigmaDat - PD.LoIndx.Point.SigmaDat +1) pre-computed error (quasi) ellipsoids. We provide a conversion factor from the standard deviation error ellipsoid to PD.DatDscrptors.Uncerty(i) for lower and upper error bars for the diagonal elements of the pre-computed error (quasi)ellipsoid(s). The x

[133] We use "(quasi)" to indicate the extension to the error volume of the effect of a one dimensional non-symmetrical error probability distribution such that the most probable errors constituting a fixed percentile of errors involves upper and lower error bars that have a differing distance from the mean. So, for a principal axis of a quasi ellipse, there can be two semi-major axis values, corresponding to an upper and a lower error bar respectively.



conversion factors are stored first, the y conversion factors stored second, and the z conversion factors are stored third. For example, if PD.DatDscrptors.Uncerty(1) equals 0.9, then these conversion factors convert standard deviations to the extents of the .9P error (quasi) ellipsoid with equal probability errors at the intersections of the principal axes and the (quasi) ellipsoidal surface. Note that the following content would only be populated at Enterprise Level-2 or higher.

ES.Swath.HiConfdncP_Fac(1,3_through_[3+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat] ,1):
Converts the standard deviation uncertainty of the pre-computed ground error ellipse x extent (first diagonal element) in ES.Point.SigmaXYZ(j,k) to the lower error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.
ES.Swath.HiConfdncP_Fac(1,3 through_[3+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat],2):
Converts the standard deviation uncertainty of the pre-computed ground error ellipse x extent in ES.Point.SigmaXYZ(j,k) to the upper error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.
ES.Swath.HiConfdncP_Fac(2,3 through_[3+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat] ,1):
Converts the standard deviation uncertainty of the pre-computed ground error ellipse x extent (first diagonal element) in ES.Point.SigmaXYZ(j,k) to the lower error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.
ES.Swath.HiConfdncP_Fac(2, 3 through_[3+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat],2):
Converts the standard deviation uncertainty of the pre-computed ground error ellipse x extent (first diagonal element) in ES.Point.SigmaXYZ(j,k) to the upper error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(1,4+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat_through_[4+2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)] ,1):
Converts the standard deviation uncertainty of the pre-computed ground error ellipse y extent (second diagonal element) in ES.Point.SigmaXYZ(j,k) to the lower error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.
ES.Swath.HiConfdncP_Fac(1,4+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat through_[4+2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)],2):
Converts the standard deviation uncertainty of the pre-computed ground error ellipse y extent in ES.Point.SigmaXYZ(j,k) to the upper error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.
ES.Swath.HiConfdncP_Fac(2,4+ PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat through_[4+ 2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)] ,1):
Converts the standard deviation uncertainty of the pre-computed ground error

ellipse y extent (first diagonal element) in ES.Point.SigmaXYZ(j,k) to the lower error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(2, 4 + PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat through [4+ 2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)],2):

Converts the standard deviation uncertainty of the pre-computed ground error ellipse y extent (second diagonal element) in ES.Point.SigmaXYZ(j,k) to the upper error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(1,5+ 2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat) through [4+2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)] ,1):

Converts the standard deviation uncertainty of the pre-computed ground error ellipse z extent (third diagonal element) in ES.Point.SigmaXYZ(j,k) to the lower error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(1,5+ 2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat) through [5+3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)],2):

Converts the standard deviation uncertainty of the pre-computed ground error ellipse z extent in ES.Point.SigmaXYZ(j,k) to the upper error bar of the PD.DatDscrptors.Uncerty(1) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(2,5+ 2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat) through [5+ 3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)] ,1):

Converts the standard deviation uncertainty of the pre-computed ground error ellipse z extent (third diagonal element) in ES.Point.SigmaXYZ(j,k) to the lower error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(2, 5 + 2*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat) through [5+ 3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat)],2):

Converts the standard deviation uncertainty of the pre-computed ground error ellipse z extent (third diagonal element) in ES.Point.SigmaXYZ(j,k) to the upper error bar of the PD.DatDscrptors.Uncerty(2) measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(1, 6+3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat,1): Converts the standard deviation uncertainty of the Mueller matrix elements to the lower error bar of the PD.DatDscrptors.Uncerty(1) polarization filter measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(1, 6+3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat,2): Converts the standard deviation uncertainty of the Mueller matrix elements to the upper error bar of the PD.DatDscrptors.Uncerty(1) polarization filter measurement range referenced to the mean.

ES.Swath.HiConfdncP_Fac(2, 6+3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat,1): Converts the standard deviation uncertainty of the Mueller matrix elements to the lower error bar of the PD.DatDscrptors.Uncerty(2) polarization filter measurement range referenced to the mean.
ES.Swath.HiConfdncP_Fac(2, 6+3*(PD.HiIndx.Point.SigmaDat-PD.LoIndx.Point.SigmaDat,2): Converts the standard deviation uncertainty of the Mueller matrix elements to the upper error bar of the PD.DatDscrptors.Uncerty(2) polarization filter measurement range referenced to the mean.

Populating any element of ES.Swath.HiConfdncP_Fac(i,j,k) with a negative number indicates that the value is N/A and should be ignored.

Governing quality metrics: The accuracy of ES.Swath.HiConfdncP_Fac is parameterized by the number of percent shortfall compared to PD.DatDscrptors.Uncerty(j) for which the value of the product of the PD.DatDscrptors.Uncerty(j) factor and the standard deviation spans the statistical return interval for large enough sample that the PD.DatDscrptors.Uncerty(j) value is stable to within 10%. Thus, a quality metric value of 0% or less is good, indicating that all PD.DatDscrptors.Uncerty(j) or more of the required photon roundtrip times to a fixed range occur within ES.Swath.HiConfdncP_Fac times the matching standard deviation uncertainty. If PD.DatDscrptors.Uncerty(1) =.95, a quality metric of 20% would indicate that only 75% of the photons actually return in the interval given by the value of ES.Swath.HiConfdncP_Fac times the matching standard deviation uncertainty, and this would be a low quality value for the ES.Swath.HiConfdncP_Fac .

See Also: ES.Swath.GM_SigmaT SC.Swath.SigmaClkStep to PD.HiIndx.Swath(13) PD.DatDscrptors.Uncerty(j) ES.Point.SigmaXYZ(j,k)

20.2.159 ES.Swath.PSF(j,k) [L1,2,3,4,5]

Brief Definition: ES.Swath.PSF(j,k) indicates the spatial size of a focal plane element's angular resolution cell in channel, j, at the several ranges in the scene for a collection.

indices: Index, j, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set.
Index, k, = 1,2,3,4

Units/representation	Meters, single float
Treatment/usage	An optical element's diffraction limited spatial response is described by the Point Spread Function (PSF). This is generally a two dimensional function in angle space and, in the far field, it is independent of range. Projecting this function to a surface orthogonal to the beam at a nominal range for a dataset results in level contours that are approximately elliptical[134]. ES.Swath.PSF is the geometric mean of the major and minor axes of the projected -3dB PSF level on the beam normal surface. This distance in meters is provided for a sensor to target range characteristic of k=1, the longest ranges in the data set. k=2, the shortest ranges in the set k=3, the approximately mean range in the data set. Note that for high volume optical systems using lenses and mirrors, it is not practical to produce a GSD that is smaller than the diffraction limited Airy disc. Doing so would require devoting many imaging elements to the area of a diffraction limited return, which comes at the expense of area coverage. A lidar intended for producing VRP data will not employ such a strategy in an operational collection mode. While the ratios may vary in a multiple wavelength system, only the lowest ratio of GSD to point spread function is of interest and that would determine the choice of value posted to ES.Swath.PSF(j,k). Users and tools can then bound the oversample ratio and compare it to target scene element sizes of interest to generate confidence metrics and detection predictions. ES.Swath.PSF(j,4) carries the effective receiver entrance aperture diameter for channel, j. The term, "effective", here accounts for the reduction in the aperture light collecting ability due to optics issues such as vignetting, aperture weighting due to coatings, compound lens functioning, etc. This aperture size would apply to a photon link budget. The effective *resolving* aperture diameter is derivable from the PSF values in k=1,2,3, the (center) wavelength for each channel, and the corresponding ranges in MD.Range.*.
Governing quality metrics	
See Also	MD.Qual.GSD MD.Range.*

[134] For common instrument and collection designs, the ellipse is nearly circular.

ES.Truth.*

ES.Truth.* parameters describe collected truth targets intended for positional or radiometric calibration support.

20.2.160 ES.Truth.IllusKMZ.*j* [L1,2,3,4,5]

Brief Definition	ES.Truth.IllusKMZ.*j* provides an illustration of the local region about ground truth points in the lidar data set.
indices	None However, the evaluated string, *j*, behaves much like an index. A separate instantiation of the base tag name ES.Truth.IllusKMZ is created for each value of j. The substring *j* is most conveniently given values that are sequentially numeric characters so that the ordered values of *j* visually correspond to the value of the index, j, in the list of character string lengths provided in PD.Wordsize.Truth(j). Thus *j* would range over the BCS values of the illustration index. This supports reducing data volume for a 'logical array' of characters strings of potentially largely different lengths.
Units/representation	Dimensionless, BCS Character, string length is variable and provided by the values of PD.Wordsize.Truth(j)
Treatment/usage	Every value of ES.Truth.IllusKMZ.*j* is a zipped KML text of truth feature(j), which provides graphic illustration of the scene indicating how to recognize the truth points contextually according to their surroundings. A 3-D graphic in zipped KML acts as a diagram when interpreted by a KML browser, such as Google Earth. This illustration can be interpreted according to the "Recognition Code"supplied by MD.Truth.Dat(3).
Governing quality metrics	
See Also	PD.Wordsize.Truth(j) MD.Truth.Dat(3)

20.2.161 ES.Truth.CalChRatios(ich) [L2,3,4,5]

Brief Definition	ES.Truth.CalChRatios(ich) provides the factors that convert individual channel intensity data into relative radiometric channel intensity data.
indices	Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m), that is, over the channels of the current data set.
Units/representation	Dimensionless, single Float
Treatment/usage	The purpose of ES.Truth.CalChRatios(ich) is to provide calibration coefficients so that channel intensity data stored as integers can be converted to relative radiometric units such that the lowest channel index

has a calibration ratio of unity. Population of ES.Truth.CalChRatios(ich) serves two objectives: 1) supporting multichannel exploitation, and 2)supporting management of byte volume.

Per the discussion of ES.Truth.IntnsyRef(itp,ich,j), intensity data is first created in channel specific units. That is, channel 1 data is formed in channel 1 units, channel 2 data is formed in channel 2 units, and so on. In order to convert the intensity of a channel of higher than the lowest channel index, to the units of the channel of the lowest channel index, we multiply the "measured" intensity ratio by the ratio of channel 1 units to channel ich units.

Consider a 2 channel system for illustration. The relative calibration is achieved by expressing channel 2 measurements in channel 1 units, even when neither are known in an absolute sense. To achieve this, we identify the points in the point cloud that were collected at the position of the calibrated truth targets. Consider a truth target at the position indicated by "itp"=1. Initially there will be a channel-1 measurement set (channel 1 intensities for all the points in the point cloud on the target at position "ipt") and a channel-2 measurement set (likewise, the channel 2 intensities for all the points in the point cloud on the target at the same position "ipt"). These measurements sets will likely have some variation in the intensities in a single channel due to speckle and system noise. After some procedure[135] reduces that data set, we will have a number for the intensity of the truth target in each channel. (And also, we will have a number for the uncertainty for the intensity in each channel.) The value of the intensity in channel 1 in channel 1 units is the output of the statistical estimation process and we need to do nothing more with that number. We now turn our attention to changing the value output from the statistical estimation process in channel 2 from channel 2 units to channel 1 units. To be more specific,
Let us call the ratio (channel 2 counts in channel 1 units)/(channel 1 counts in channel 1 units) the "true ratio".
Let us call the ratio (channel 2 counts in channel 2 units)/(channel 1 counts in channel 1 units) the "measured ratio".
Then the "true ratio" is found from the "measured ratio" by multiplying by the ratio (Ch1 units/Ch2 units) at any point in the point cloud.
We define ES.Truth.CalChRatios(ich)= (Ch1 units/Ch2 units).

[135] This procedure is up to the using application .

For original collection channels, the calibration ratios can be found from the point cloud intensities and the truth data, ES.Truth.IntnsyRef(itp,ich,j). At truth point 1, we have
(True ratio 1)/(measured ratio 1)=(Ch1 units/Ch2 units)$_1$
At truth point 2, we have
(True ratio 2)/(measured ratio 2)=(Ch1 units/Ch2 units)$_2$
… and so on.

For a linear responding lidar, the (Ch1 units/Ch2 units)$_i$ should be samples of a single valued function= ES.Truth.CalChRatios(2)[136]. The process of combining the (Ch1 units/Ch2 units)$_i$ to form ES.Truth.CalChRatios(2) is up to the processor and will be some form of (confidence weighted) averaging or fitting. If the lidar is not linear in channel balance with respect to intensity level, ES.Truth.CalChRatios(ich) would be considered to be the term fac_0 in the expression,
$fac= fac_0 \times [1 + slope\cdot(X-1) + rate^{137}\cdot(X-1)^2]$ where X=Ch1 counts/Ch1ref and Ch1 ref is the value of the Ch1 counts for which $fac= fac_0$.

For as-processed channels that result from a combination of the original collected channels, ES.Truth.CalChRatios(ich) values would need to be modified so that they represent the ratio, (Ch1 units/Ch2 units) <u>after generation of the new channels</u>. In this way, we can convert stored channel intensities at enterprise L-3 or higher to relative radiometric values.

Governing quality metrics

See Also: ES.Truth.IntnsyRef(itp,ich,j) PD.HiIndx.Channlzn.ChRg(m) PD.HiIndx.Channlzn.Rg_m

20.2.162 ES.Truth.IntnsyRef(itp,ich,j) [L1,2]

Brief Definition: ES.Truth.IntnsyRef(itp,ich,j) provides spectral band and polarization specific reflectivity for truthed points in the scene of the lidar data set.

indices: Index, itp, ranges from PD.LoIndx.Truth(1) to PD.HiIndx.Truth(1), that is generally from 1 to the number of truthed targets points documented for the data set.

[136] We illustrate with channels indexed by ich=1,2. Coders may choose to start the channel indexing at 0.
[137] We defer fully defining the slope and rate terms as metadata until a lidar system with non-linear relative channel calibration emerges. For now, the linear calibration assumes that slope and rate are zero.

	Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. Index j=1,2
Units/representation	Dimensionless, single Float
Treatment/usage	ES.Truth.IntnsyRef(itp,ich,j) is defined for the purpose of supporting relative radiometric calibration of collected multichannel lidar data. For each value of truth point index, itp, ES.Truth.IntnsyRef(itp,ich,j) provides channel specific reflectivity data for backscattering in the incidence angle range of the data collection.[138] The channel designated as the having the lowest channel index (usually 0 or 1) in SC.Channlzn.Descr(j,k) will always have a channel specific relative reflectivity of unity. We choose to populate this as a data item for simplicity, although one does not need the metadata value as this is defined in this paragraph. This avoids special case indexing and knowledge of this fact can always be used by applications that ingest this truth data array. The amount of data affected is relatively small, so this is not a substantial data volume management issue. For truthed point "itp" and original collection channel, ich: ES.Truth.IntnsyRef(itp,ich= lowest channel,1)= 1.000E+00 ES.Truth.IntnsyRef(itp,ich= lowest channel,2)= 0.000E+00 ES.Truth.IntnsyRef(itp,ich= any higher channel number,1)= (reflectivity of channel ich)/(reflectivity of lowest channel) ES.Truth.IntnsyRef(itp,ich= any higher channel number,2)= uncertainty in the populated value of ES.Truth.IntnsyRef(itp,ich= any higher channel number,1) Note that the relative reflectivity ratios provided by ES.Truth.IntnsyRef(itp,ich= any higher channel number,1) may be either greater than or less than 1.000E+00. Multichannel intensities can be reported on a non-normalized or self normalized basis. Non-normalized channel intensity reports the "raw" counts (recall that the intensity data are integers) for recorded or derived intensities. Normalization might be imposed by making the minimum

[138] This puts the burden of the selection of angle dependent reflectivity ratios to report in the metadata on the operator to design the reference truth targets and their laboratory or 'in situ' calibrated measurement procedure so that the data is relevant to the collection scenario. In particular, knowledge of the likely incidence angle on objects of interest may be needed to select the correct data to populate into the ES.Truth.IntnsyRef(itp,ich,j) data item.

increment a value of 1, or perhaps the maximum value might be made to be the maximum value that the integer byte size can support. Effectively, the native intensities of points determined by the data for the channel in which they are measured without regards to other channels or a'priori calibration data, are expressed in the units of the channel of measurement. Simply, we begin with channel 1 data measured in channel 1 units and channel 2 data measured in channel 2 units, and so on.

The observed channel responses over calibration targets are to be combined with the a'priori enterprise L-1 data in ES.Truth.IntnsyRef(itp,ich,j) to produce calibration ratios (which can be stored in ES.Truth.CalChRatios(ich)) for use in level-3 and higher data.

If the truthed targets have no radiometric calibration data, ES.Truth.IntnsyRef(itp,ich,j) should not be populated. If only some of the truthed targets have radiometric calibration data, then those values of ipt that have **no** radiometric calibration truth data should be set as follows:

ES.Truth.IntnsyRef(itp,ich,1)=0.0E+00 and
ES.Truth.IntnsyRef(itp,ich,2)=1.0E+10.

Governing quality metrics

See Also PD.HiIndx.Truth(1) ES.Truth.CalChRatios(ich)

20.2.163 ES.Truth.Pos(itp,j) [L1,2,3,4,5]

Brief Definition ES.Truth.Pos(itp,j) provides geodetic coordinates for truthed points in the scene of the lidar data set.

indices Index, itp, ranges from PD.LoIndx.Truth(1) to PD.HiIndx.Truth(1), that is generally from 1 to the number of truthed points documented for the data set.
Index, j, =1,2,3

Units/representation For j=1,2 decimal degrees, for j=3 meters, single Float

Treatment/usage For each value of truth point index, itp, ES.Truth.Pos provides the coordinates of a geodetic position. ES.Truth.Pos(itp,j) gives the earth frame coordinates of each truth point.
ES.Truth.Pos(itp,1) gives the latitude in decimal degrees. Positive values are in the Northern hemisphere and negative values are in the Southern hemisphere.
ES.Truth.Pos(itp,2) gives the longitude in decimal degrees. This is measured from the IERS Reference Meridian, also called the International Reference Meridian, which is the reference meridian (Prime Meridian, 0° longitude) for GPS. Positive values are measured Eastward and negative

values are measured westward.
ES.Truth.Pos(itp,3) is the height or altitude[139] above the geoid (follows kml convention). The specific geoid reference is provided by ES.Origin.GeodtcEpoch(2). A small negative value corresponds to being slightly below the geoid (mean sea level surface). This metric measures distance from the geoid in the ellipsoid normal direction. This is the "normal" height which is referenced to the geoid and measured along the ellipsoid normal.

Governing quality metrics	
See Also	PD.HiIndx.Truth(1) ES.Origin.GeodtcEpoch(2)

20.2.164 ES.Truth.CovCorrs(itp, j) [L1,2,3,4,5]

Brief Definition	ES.Truth.CovCorrs (itp,j) provides the position error correlations for the positions of the truth points provided in ES.Truth.Pos(itp,j).
indices	Index, itp, ranges from PD.LoIndx.Truth(1) to PD.HiIndx.Truth(1), that is generally from 1 to the number of truthed points documented for the data set. Index, j, =1,2,3
Units/representation	Dimensionless, single Float
Treatment/usage	ES.Truth.CovCorrs(itp,j) supports a separate covariance matrix for each value of itp, the truth point index. Let subscript, 1, correspond to easting, subscript,2 correspond to northing, and subscript 3 correspond to local vertical, Z in meters. Then the error cross correlations in the truthed positions are: ES.Truth.CovCorrs(itp,1) = ρ_{12}, ES.Truth.CovCorrs(itp,2) = ρ_{13}, ES.Truth.CovCorrs(itp,3) = ρ_{23}, Note that we require that the covariances on a truth point to truth point basis must be free of correlation such that a multi-point covariance for the truth points collectively can be constructed from the data at each point separately. This weak constraint on the ground truthing process avoids the need to provide a metadata framework for storing multiple point covariances for an unknown number of points or presenting an artificial model for naturally uncorrelated measurements in the absence of fore

[139] For points elevated above the ground. Normal height is defined in NIMA TR8350.2 as a DoD use metric. It is the formalized height above mean sea level.

knowledge of any specific expedients that would have led to error correlations.

Governing quality metrics

See Also

20.2.165 ES.Truth.CovSigs(itp, j) [L1,2,3,4,5]

Brief Definition	ES.Truth.CovSigs(itp,j) provides the error standard deviations for the positions of the truth points provided in ES.Truth.Pos(itp,j).
indices	Index, itp, ranges from PD.LoIndx.Truth(1) to PD.HiIndx.Truth(1), that is generally from 1 to the number of truthed points documented for the data set. Index, j, =1,2,3
Units/representation	Meters, single Float
Treatment/usage	ES.Truth.CovSigs(itp,j) supports a separate covariance matrix for each value of itp, the truth point index. Let subscript, 1, correspond to easting, subscript,2 correspond to northing, and subscript 3 correspond to local vertical, Z in meters. Then the position uncertainty standard deviations are: ES.Truth.CovSigs(itp,1) = σ_1, ES.Truth.CovSigs(itp,2) = σ_2, ES.Truth.CovSigs(itp,3) = σ_3,
Governing quality metrics	
See Also	ES.Truth.CovCorrs

ES.WFClp.*

ES.WFClp.* parameters describe exploitation support for reported waveform data vectors

20.2.166 ES.WFClp.ChNum.*TileNum*(ipi,iray) [L1,2]

Brief Definition ES.WFClp.ChNum.*TileNum*(ipi,iray) provides channel identification for

waveform data.

indices

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile".

Use of *TileNum* for wave form clips enterprise L-2:

Waveforms in enterprise L-2 data are subsets ("clips") of the full receive window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "1R*nnnnnn*" needs to be unique.

Index, ipi, ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*. Index, ipi, is a consecutively valued index counter that counts only the "ip" pulse values for which waveforms are presented. But ipi is not the pulse number itself unless all pulses are consecutively listed.

ip	ipi	Has a waveform vector
1		no
2	1	yes
3	2	yes
4		no
5	3	yes
6		no

The above table is effectively a representation of PD.WFClp.PulseNum.*TileNum*(ipi[140]), which assigns an index, ipi, to each entry in the list of "used pulses" with populated waveform clips in a tile by *TileNum*, which is specific to a return number.

Index, iray, ranges from PD.TileMap.RayRange(j_tile, 1) to PD.TileMap.RayRange(j_tile, 2), that is from the first ray in the tile to the last ray in the tile. The definitions of iray, if populated, will either be present in

[140] The index, "ipi", replaces "ip" at L-2 so that sparse pulse wave form population is possible. At L-1, every pulse wave form (RxDatVec) must be in the file so an ip that just counts IPPs will not waste storage. But only a small subset of pulses may have waveforms at L-2, so ipi is not a consecutive counter of IPPs, only of "used IPPs

SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) or SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). Enterprise level-1 will tend to use the global iray definitions. Enterprise level-2 data sets will vary depending on the level-2 generating application and the determination of whether there are few enough waveforms to warrant using a small, tile oriented, waveform list approach to minimize file size.

Special note for waveform data resulting from channel combination: In the case that channel combination has been used to create the wave form clips at enterprise level-2, a situation akin to synthetic point generation arises and there is not necessarily a single pulse or ray that identifies a waveform vector any longer. However, we shall identify a channel combined waveform clip with a time within the temporal integration interval and a look direction within the field of view of integration referenced to that time. The pulse and ray that most closely correspond to the time and direction attributed to the channel combined waveform clip have index values defined on a *collected* channel basis. Those are used to identify the channel combined waveform data. Thus, the translation of ip and iray index values to times and directions is exactly the same for channel combined data as for collected channel data. Only the interpretation must be tempered by the above comments and considered in light of PF.Channlzn.Hist.WF_Dat(j).

Units/representation
Dimensionless, UInt8

Treatment/usage
Within a single tile, each waveform data vector, LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) is reported for a specific *collection* channel at enterprise level-1, but may be a processed channel at level-2. ES.WFClp.ChNum.*TileNum*(ipi,iray) provides the channel identification for waveform data. We refer to collection channels, not derived channels created by processing, when discussing received wave form data that have not been subject to channel combination. Channel combination of waveform data is permitted (but not necessarily performed) at enterprise level-2, but not level-1. A channel identification is the value of j identifying the pertinent row of channel defining parameter values in SC.Channlzn.Descr(j,k) (for enterprise level-1) or PF.Channlzn.Descr(j,m) (for enterprise level-2) .

Note that for ES.WFClp.ChNum.*TileNum*(ipi,iray), the use of the index reference “ipi” makes this general for both enterprise L-1 and L-2. At L-1, ipi is the same as ip, but at L-2, we only care about the ipi, which may be a more limited set of index values due to only putting a fraction of the collected waveforms into the L-2 product. Thus the channel for a waveform vector can be determined by a user of a data set for a known specific tile of origin

without needing to translate ipi to ip.

The generator of a data set with a populated ES.WFClp.ChNum.*TileNum*(ipi,iray) will need to know that the tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter, ip, starts over from 0 for each tile. The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts, 3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases)
Care must be taken by the data set writing software to assure that the values of ip and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.

Governing quality metrics

See Also

PD.WFClp.PulseNum.*TileNum*(ipi) SC.WF.RayDesc.ArryGlbl(iray) SC.WF.RayDesc.ElmtGlbl(iray) LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) PF.Channlzn.Descr(j,m) SC.Channlzn.Descr(j,k)

20.2.167 ES.WFClp.Locs.*TileNum*(ipi,iray, j) [L2]

Brief Definition

ES.WFClp.Locs.*TileNum*(ipi,iray, j) provides location information about the return associated with the waveform clip identified by pulse, ipi, and ray, iray.

indices

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile".

Use of *TileNum* for wave form clips enterprise L-2:

Waveforms in enterprise L-2 data are subsets ("clips") of the full receive

window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "1R*nnnnnn*" needs to be unique.

Index, ipi[140], ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*.

Index, iray, ranges from PD.TileMap.RayRange(j_tile, 1) to PD.TileMap.RayRange(j_tile, 2), that is from the first ray in the tile to the last ray in the tile. The definitions of iray, if populated, will either be present in SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) or SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). Enterprise level-1 will tend to use the global iray definitions. Enterprise level-2 data sets will vary depending on the level-2 generating application and the determination of whether there are few enough waveforms to warrant using a small, tile oriented, waveform list approach to minimize file size.

The index j =1,2,3

Units/representation: Dimensionless, variable[141] Integer, number of bytes found in PD.Wordsize.Pos

Treatment/usage: ES.WFClp.Locs.*TileNum*(ipi,iray, j) is intended for use in enterprise level-2 lidar products composed of a combination of a point cloud and a supplement of waveform clips, where the number of waveform clips is far smaller than the number of pulses used to collect the data for the overall lidar product. For a pulse indicated by the value of ipi [which can be translated to a pulse number via the table, PD.WFClp.PulseNum.*TileNum*(ipi)] and the ray indicated by the value of iray, ES.WFClp.Locs.*TileNum*(ipi,iray, j) provides the xyz position implied for the first fast time sample of the waveform clip by

[141] For HDF5 transport, the DAL can provide the integer wordsize without a separate metadata item for the number of bits/bytes (because this is specifically support by the HDF5 software). But the general case would need a metadata item that gives the wordsize if it is variable.

assuming that photons follow the ray incident at the detector element associated with the value of iray. That position is computed via the time to range conversion alglorithm as if no scattering occurs.
ES.WFClp.Locs.*TileNum*(ipi,iray, 1) = x coordinate step number
ES.WFClp.Locs.*TileNum*(ipi,iray, 2) = y coordinate step number
ES.WFClp.Locs.*TileNum*(ipi,iray, 3) = z coordinate step number
The offsets to convert position indices to Cartesian coordinates in meters are in ES.Offset.X(j_tile), ES.Offset.Y(j_tile), and ES.Offset.Z(j_tile). The value of j_tile to use is the one that corresponds to *TileNum* as provided in PD.TileMap.TileNum(j_tile). In essence, ES.WFClp.Locs.*TileNum*(ipi,iray, j) is treated just as point positions in a point cloud found in LD.Point.Pos.*TileNum*(ipt,j)

For returns higher than 1, this may not be a realistic assumption for determining the actual scene scattering position, but it is adequate to approximately locate the region from which the waveform clip has been scattered. The principle use case for this data is that a user will identify a volume of interest in a point cloud and wish to retrieve waveform clips from that specific region. The available waveform clips that are closest to the position of the volume defined in a data query intended to discover and retrieve waveform clips of interest can be determined by applying the Pythagorean theorem to the XYZ values in ES.WFClp.Locs.*TileNum*(ipi,iray, j).
The integer values populated into ES.WFClp.Locs.*TileNum*(ipi,iray, j) must be converted to coordinates in meters via the same scaling and offset transformation process used to convert point cloud locations LD.Point.Pos.*TileNum*(ipt,j) to meter valued Cartesian coordinates. The scale factors to be used are provided globally for the given data set in ES.Scale.X_Fac et al and the offsets are in ES.Offset.X(j_tile) et al.

The tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter ip starts over from 0 for each tile. The term, “tile level”, indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = “tile level” ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3)
where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point

oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases)
Note that the encoding of the return number into *TileNum* restricts all wave form clips within a tile to have the same return number. But the locations of waveform clips as documented in ES.WFClp.Locs.*TileNum*(ipi,iray, j) may be within the same small spatial regions in different tiles. When using the location data to find clips, it must be done on a single return basis. See discussion of fully qualified queries in 10.2.1.2.

Care must be taken by the data set writing software to assure that the values of ip and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.

Governing quality metrics

See Also

ES.Scale.X_Fac ES.Offset.X(j_tile) PD.HiIndx.WFClpPulseNum.*TileNum* PD.WFClpPulseNum.*TileNum*(ipi) PD.TileMap.TileNum(j_tile) LD.Point.Pos.*TileNum*(ipt,j) PD.Wordsize.Pos

20.2.168 ES.WFClp.RtnsCode.*TileNum*(iray/irays, ipi/j) [L2]

Brief Definition

ES.WFClp.RtnsCode.*TileNum*(iray, ipi[140]) encodes information about the return associated with the waveform clip identified by pulse, ipi, and ray, iray.

indices

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile".

Use of *TileNum* for wave form clips enterprise L-2:

Waveforms in enterprise L-2 data are subsets ("clips") of the full receive window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the

digits after "R" just count tiles and only the entire "1R*nnnnnn*" needs to be unique.

Default case: ES.WFClp.RtnsCode.*TileNum*(**iray, ipi**) is an array of return codes in rows indexed by the ray number and columns indexed by the tile specific ordinal pulse value, ipi.
{PD.IntTileMap.RtnsCode.*TileNum*.Flg unpopulated or explicitly set to 0}

Index, ipi, ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*. Index, ipi, is a consecutively valued index counter that counts only the "ip" pulse values for which waveforms are presented. But ipi is not the tile pulse number itself unless all pulses are consecutively listed.
The tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter ip starts over from 0 for each tile. The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases)
Index, iray, ranges from PD.TileMap.RayRange(j_tile, 1) to PD.TileMap.RayRange(j_tile, 2), that is from the first ray in the tile to the last ray in the tile. The definitions of iray, if populated, will either be present in SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) or SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). Enterprise level-2 data sets will vary in global vs. tile ray definition depending on the level-2 generating application and the determination of whether there are few enough waveforms to warrant using a small, tile oriented, waveform list approach to minimize file size.

Repetition factor in use case:
ES.WFClp.RtnsCode.*TileNum*(**irays, j**) is an array of return codes in rows indexed by the distinct pulse-ray cases[142] of a specific return code. This indexing supports repetition factor use to reduce data volume. {PD.IntTileMap.RtnsCode.*TileNum*.Flg explicitly set to value greater than 0} See internal tile mapping.

An intervening new value of return code followed by a string of return codes with a value that was higher up the list sets off a second distinct pulse ray case set from the first due to a break in the run of return code values. Sorting the data by the return code value itself would eliminate repeated return codes in separate cases because they would all have been run together by the sorting process. It may be preferable, however, to sort by pulse number (temporal sorting) [143]. Though this too can create long runs of return code repeats, it permits that multiple blocks of the same return code can occur. The trade is a little less reduction in total data volume for being able to support some process that may benefit from storing time ordered waveforms.

Index, irays, ranges from a start value (generally 0 or 1 depending on the coding style to (PD.IntTileMap.RtnsCode.*TileNum*.Flg-1) or PD.IntTileMap.RtnsCode.*TileNum*.Flg so that there are PD.IntTileMap.RtnsCode.*TileNum*.Flg consecutive integer values of the index, irays. The index, irays, thus ranges over the number of pulse-ray cases for the tile and that number is found in PD.IntTileMap.RtnsCode.*TileNum*.Flg

The ranges of ip and iray in the logical, but sparse, array are already compressed by using ipi in place of ip. But when there are extended runs of the same value of the return code for a sequence of pulse-rays, we can reduce the data array substantially by ordering the pulse rays according to the values of the return code and indicating the number of pulse rays in a run that have a particular value of return code. Knowledge of the sequence permits reconstruction of the return code to pulse-ray assignment for any pulse ray with a waveform reported.
Effectively, the logical array ES.WFClp.RtnsCode.*TileNum*(iray, ipi) for the

[142] A "pulse-ray case" is a run a pulse-rays that have the same value of return code and can thus be represented by a number of consecutive repeats and the return code value. The trace to the identity of specific pulse rays is possible due to the ordering of the pulse-ray references before reduction to repeat factor form and the presence of the ray number in the j=1 slot of the ES.WFClp.RtnsCode.*TileNum*(**irays,j**) form of the array.

[143] These are available as options since we have to translate the ipi to ip in enterprise level-2 data sets that have waveform clips regardless of the sorting order of the waveform clips in the tile.

tile indicated by *TileNum* is used to create a row major based linear index that effectively treats the matrix of pulse rows and ray columns as a column vector.

The following table illustrates an alternate view of the ES.WFClp.RtnsCode.*TileNum*(iray, ipi) array information:

Column vector of Pulse-rays Column vector of return codes

Pulse-ray number	**Return code**
1	Value from 0 to 196, 255
2	Value from 0 to 196, 255
…	Value from 0 to 196, 255
Last pulse-ray for tile	Value from 0 to 196, 255

There is one value of "irays" for each run of consecutive repeated values of the return code in the above table. That run will have a length of some number of pulse-rays. See the following example.

Pulse-ray number	**Return code**	**Pulse ray case**	**Ordinal value of pulse ray in the pulse ray case**
1	1st return code value	1	1
2	1st return code value	1	2
3	1st return code value	1	3
…	1st return code value	1	49
50	2nd return code value	2	1
51	2nd return code value	2	2
…	2nd return code value	2	151
201	1st return code value	3	1
202	1st return code value	3	2
…	1st return code value	3	399
600	3rd return code value	4	1
…	3rd return code value	4	2
Last pulse-ray for tile	3rd return code value	4	Last pulse-ray for tile-600 +1

The following table would be an unambiguous reduction of the above table if we knew the constant number of pulses in the pulse-ray case to pulse-ray translation table.

Return code	**Pulse ray case**	**Repeats in pulse ray case**
1st return code value	1	49
2nd return code value	2	151
1st return code value	3	399
3rd return code value	4	Last pulse-ray for tile-600 +1

So that we can handle pulse-ray cases that have different numbers of pulses and rays each, we augment the table with the (tile specific, when appropriate) ray index for the pulse-ray case. We can suppress the pulse-ray case since that is merely the row of the table.

Ray number (iray value)	**Repeats in pulse ray case**	**Return code**
$iray_1$	49	1st return code value
$iray_2$	151	2nd return code value
$iray_3$	399	1st return code value
$iray_4$	Last pulse-ray for tile-600 +1	3rd return code value

Index, j, =1,2,3 (or possibly 0,1,2, depending on the software coding convention) This corresponds to the columns of the above table.

Treat the repetition factor bearing array as 2-D with ES.WFClp.RtnsCode.*TileNum*(irays,j) populated as follows”

That is, ES.WFClp.RtnsCode.*TileNum*(irays,1), is the ray number for each pulse-ray case, which is in turn indexed by irays. ES.WFClp.RtnsCode.*TileNum*(irays,2) is the number of times the value in ES.WFClp.RtnsCode.*TileNum*(irays,3) consecutively repeats in a single stretch of “ipi” values corresponding to ‘no change’ over a range of pulses. In other words, ES.WFClp.RtnsCode.*TileNum*(irays,2) is the number of consecutive pulse values for a ray given by ES.WFClp.RtnsCode.*TileNum*(irays,1) for which the value of return code is given in ES.WFClp.RtnsCode.*TileNum*(irays,3). ES.WFClp.RtnsCode.*TileNum*(irays,3) is the return code, not return number, for the waveform clip described for ray index, ES.WFClp.RtnsCode.*TileNum*(irays,1). The pulse number, (ipi value part of the pulse-ray case definition) is to be recovered by accumulating the repeat factors in ES.WFClp.RtnsCode.*TileNum*(irays,2) per the discussion in

10.2.1.5. This is consistent with the treatment of PD.WFClp.Nsamps.*TileNum*(iray,ipi).

Units/representation Dimensionless, UInt8

Treatment/usage The encoding of the returns information for waveform clips, for simplicity, uses the same list of values as for points. Partial waveform clips will only be provided near (possibly extended) peaks, since it makes sense to only report a clean return spike as a point and makes little sense to report non-return (i.e. just noise) data in enterprise level-2 data sets. However, for a point, the only source of information as to what the return number is comes via the returns code. For waveforms, the return number is encoded into the *TileNum*. Thus, the return number in the returns code is redundant information for waveform clips. However, since the reality of focal plane sensitivity, scattering and propagation physics limits the practical number of returns for a single pulse ray to just a few, the restriction to the same number of returns for wave clips as with points is not truly limiting. The additional information of how many earlier returns were reported as points is of potential interpretative value. Since the sum of point returns and wave returns is the total number of returns [per the appendix discussion of waveform clip considerations, we do not permit reporting both waveforms and points for the same PCD return number in the same product], separate population of a returns code for waveforms permits clarifying whether a clip might be preceded by some points, or some other clips.

Lidar returns may result from decidedly point-like linear mode wave-form peaks, or from more spread out linear mode returns[144]. Depending on the transmitted power and receive window in linear mode, multiple returns for a single pulse may be recorded in the lidar sensor. Thus, there is information to be relayed regarding which of the possible returns for a pulse a particular waveform clip represents. Also, the total number of returns reported for the same pulse-ray, and how many of the preceding returns for a given pulse-ray were point-like can be of value in interpreting the lidar measurements of a scene. The following table (repeated from ES.Point.RtnsCode.*TileNum*(ipt)) shows how the return number, number of lower numbered returns that are point-like and the total returns for the same pulse-ray can be encoded into a single 8 bit word for up to 7 returns. This is an embedded Excel spread sheet that can be opened by double clicking and then scrolled down to the last code, 196.

[144] This suggests that when appropriate, wave forms may be used to characterize spatially extended returns that are distinctly not signatures of isolated points or opaque surfaces normal to the look direction of the lidar sensor.

Code	return#	lower returns that are points	Total returns
0	0	0	0
1	1	0	1
2	1	0	2

It is recommended that programs using the return codes for waveform clips adopt the convention that a return code of 255 represents an invalid case to indicate that a return code has not been supplied for a waveform clip, or for a pulse and ray combination that does not have a wave form clip at all in the data set. It is permissible to create a *TileNum* for an enterprise L2 data set waveform tile that "builds in" a return number greater than 7. Such a return number has no valid return code and ES.WFClp.RtnsCode.*TileNum*(iray, ipi) should be left unpopulated. The waveforms can be read in, but the information about returns in the return code in the right two columns of the above table defining the return code will not be available and that can be indicated by the use of return code 255.

Care must be taken by the data set writing software to assure that the values of global IPP number, ip, and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.

Governing quality metrics

See Also

PD.WFClp.PulseNum.*TileNum*(ipi)
ES.Point.RtnsCode.*TileNum*(ipt) PD.WFClp.Nsamps.*TileNum*(iray,ip)
PD.IntTileMap.RtnsCode.*TileNum*.Flg PD.TileMap.PulseRange(j_tile,1)
LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)

20.3 PF.* Product Formation

PF.Adjustmts.*

PF.Adjustmts.* parameters describe registration derived coordinate transforms needed to properly mosaick smaller collected and processed pieces of a large point cloud into a rigid internally consistent global sampling of the scene space.

20.3.1 PF.Adjustmts.Artifacts(j) [L2,3,4,5]

Brief Definition	PF.Adjustmts.Artifacts(j) provides indication of the effectiveness of remedies for known data artifacts
indices	Index, j, ranges from PD.LoIndx.Artifacts to PD.HiIndx.Artifacts, that is over the known list of data issues
Units/representation	Dimensionless, UInt8
Treatment/usage	This is a stub for a list that corresponds to SC.Artifacts.List(j) and SC.Artifacts.Descr(j). This tells users how well any processing mitigations of known data issues, if applied, are expected to have worked. Note that when an artifacts registry is established, the alignment of j values in populated instances of PF.Adjustmts.Artifacts(j) will correspond only to listed artifacts described in SC.Artifacts.Descr(j). PF.Adjustmts.Artifacts(j)=0 --- no effort to correct artifact made PF.Adjustmts.Artifacts(j)=1 --- severity measured and considered benign, no correction made PF.Adjustmts.Artifacts(j)=2 --- Correction found, saved[145] for correction upon demand PF.Adjustmts.Artifacts(j)=3 --- Correction applied, mitigation predicted to be moderate PF.Adjustmts.Artifacts(j)=4 --- Correction applied, mitigation predicted to be very good PF.Adjustmts.Artifacts(j)=5 --- Correction applied, mitigation predicted to be so good that data can be treated as if fully remedied.
Governing quality metrics	
See Also	SC.Artifacts.List(j). PD.HiIndx.Artifacts

20.3.2 PF.Adjustmts.Mean.Pulse.*TileNum*(ip, j) [L2,3,4,5]

Brief Definition	PF.Adjustmts.Mean.Pulse.*TileNum*(ip, j) provides the aligning transformation between lidar frames when frame to frame registration is performed in the construction of scans.
indices	Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2)-1, that is, over all the pulse pairs for a tile. The first pulse pair has the same index as the first pulse and is composed of

[145] As of CMMD L2 0.9 there is no place to save corrections other than geometric items. In the interim, user defined data slots can carry these corrections, but it is likely that version(s) 1.x will be amended to address registered artifacts and their correction data when, and if, the contingency arises that there are corrections available that might either not be applied upstream or that might be reversible if adequate metadata is made available.

the first pulse and its following pulse. This labeling process can be repeated for each pulse in a tile up to the one before the last pulse. The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses.
PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.
j=1,2,3,4,5,6

Units/representation: j=1,2,3 meters, j=4,5 dimensionless, j=6, degrees[146], Float

Treatment/usage: This items is unlikely to be populated due to excessive volume except for small data sets (this why a *TileNum* substring has been added to the tag), but keeping it in the metadata dictionary keeps a structural symmetry. This item is completely analogous to PF.Scan.Adjustmts(is, j) This is the summary of adjustments needed on a frame pair basis determined by a registration process at the sub-scan level.

PF.Adjustmts.Mean.Pulse(is, 1) =earth frame x shift
PF.Adjustmts.Mean.Pulse(is, 2) =earth frame y shift
PF.Adjustmts.Mean.Pulse(is, 3) =earth frame z shift
PF.Adjustmts.Mean.Pulse(is, 4) = x-direction cosine of rotation axis
PF.Adjustmts.Mean.Pulse(is, 5) = y-direction cosine of rotation axis
PF.Adjustmts.Mean.Pulse(is, 6) = angle of rotation about about rotation axis.

The x- direction cosine measures the angle from the reference frame x-axis to the rotation axis determined in the registration process for Pulse pair, “ip”. Similarly, the y- direction cosine measures the angle from the reference frame y-axis to the rotation axis determined in the registration process for Pulse pair, is. The reference frame is indicated in PF.Registn.PulseDat(3). Unlike scans or subswaths, pulse pairs should not involve multiple tiles.
A 3 x 3 rotation matrix can be formed from the axis, angle data in elements 4,5, and 6.
$c = \cos(\theta)$; $s = \sin(\theta)$; $C = 1-c$; x= PF.Pulse.Adjustmts(is, 4)
y= PF.Pulse.Adjustmts(is, 5); θ= PF.Pulse.Adjustmts(is, 6)
$z=\mathrm{sqrt}(1-x^2-y^2)$
xs = x*s; ys = y*s; zs = z*s

[146] Chosen because we anticipate small angles and we get more precision for small angles expressed in degrees than for radians. This may not be consequential if the corrections are coarse anyway.

xC = x*C; yC = y*C; zC = z*C
xyC = x*yC; yzC = y*zC; zxC = z*xC

$$R_{ax,ang} = \begin{bmatrix} x*xC+c & xyC-zs & zxC+ys \\ xyC+zs & y*yC+c & yzC-xs \\ zxC-ys & yzC+xs & z*zC+c \end{bmatrix}$$

Governing quality metrics: Percent error

See Also: PF.Scan.Adjustments(is, j) PD.TileMap.PulseRange(j_tile,1) PF.Registn.PulseDat(3)

20.3.3 PF.Adjustmts.Mean.Scan(is,j) [L2,3,4,5]

Brief Definition: PF.Adjustmts.Mean.Scan(is, j) provides the aligning transformation between scan pairs in the creation of subswaths.

indices: Index, is, ranges from PD.LoIndx.Scan(3) to PD.HiIndx.Scan(3), that is, over the number of scans in the dataset.
Index, j= 1,2,3,4,5,6,7,8,9

Units/representation: j=1,2,3,7,8,9 Meters
j=4,5 Dimensionless
j=6 Degrees
all Single floats

Treatment/usage: This is the summary of adjustments needed on a scan pair basis determined by a registration process at the subswath level
PF.Adjustmts.Mean.Scan(is, 1) =earth frame x shift
PF.Adjustmts.Mean.Scan(is, 2) =earth frame y shift
PF.Adjustmts.Mean.Scan(is, 3) =earth frame z shift
PF.Adjustmts.Mean.Scan(is, 4) = x-direction cosine of rotation axis
PF.Adjustmts.Mean.Scan(is, 5) = y-direction cosine of rotation axis
PF.Adjustmts.Mean.Scan(is, 6) = angle of rotation about rotation axis.
PF.Adjustmts.Mean.Scan(is, 7) = ECEF x coordinate of "pin" point on the rotation axis.
PF.Adjustmts.Mean.Scan(is, 8) = ECEF y coordinate of "pin" point on the rotation axis.
PF.Adjustmts.Mean.Scan(is, 9) = ECEF z coordinate of "pin" point on the rotation axis.

The x- direction cosine measures the angle from the reference frame x-axis to the rotation axis determined in the registration process for scan pair, is. Similarly, the y- direction cosine measures the angle from the reference frame y-axis to the rotation axis determined in the registration process for scan pair,

"is". The nature of the input point blocks used in the registration that determines the adjustments provided in PF.Adjustmts.Mean.Scan(is,j) are found in PF.Registn.ScanDat(3).
A 3 x 3 rotation matrix can be formed from the axis, angle data in elements 4,5, and 6.
c = cos(θ); s = sin(θ); C = 1-c; x= PF.Adjustmts.Mean.Scan(is, 4)
y= PF.Adjustmts.Mean.Scan(is, 5); θ= PF. Adjustmts.Mean.Scan(is, 6)
$z=sqrt(1-x^2-y^2)$
xs = x*s; ys = y*s; zs = z*s (* denotes multiplication)
xC = x*C; yC = y*C; zC = z*C
xyC = x*yC; yzC = y*zC; zxC = z*xC

$$R_{ax,ang} = \begin{bmatrix} x*xC+c & xyC-zs & zxC+ys \\ xyC+zs & y*yC+c & yzC-xs \\ zxC-ys & yzC+xs & z*zC+c \end{bmatrix}$$

The "pin" point, whose ECEF coordinates are in the j=7,8,9 elements for each set of scan adjustment data, may be treated as the origin for a vector whose direction is given via the j=4, 5 elements of PF.Adjustmts.Mean.Scan(is, j).

Governing quality metrics

The pin point shall be chosen so that the mean shift over the affected scan due to the adjustment rotation is no larger than the larger of 1 percent of the length of the mean adjustment shift vector or the hard bound amount indicated in PF.Registn.ScanDat(4).

See Also

PD.HiIndx.Overlap(3) PF.Registn.ScanDat(3)

20.3.4 PF.Adjustmts.Mean.Subswath.*Iswl*(isw,j) [L2,3,4,5]

Brief Definition

PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j) provides the mean aligning coordinate transformation for subswaths at subswath layer *Iswl* in the creation of larger subswaths or the final swath.

indices

Computed substring "*Iswl*" acts somewhat like a subscript and ranges over the subswath layers, of which there are PF.Subswath.Nlayers. By making this a substring, we create an array for each subswath layer that can be kept in direct correspondence with the different length lists of adjustments in each subswath layer found in PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw,i,j). The *Iswl* substring will be a character representation of the numerical value of the subswath layer being treated by the array, PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j). In this discussion, 'iswl' is the integer corresponding to the value of the character string "*Iswl*".

Index, isw, ranges over the number of subswaths in the subswath layer indicated by the value of substring *Iswl*, that is, effectively from 1 to

PF.Subswath.NumSubswaths(iswl).
Index, j= 1,2,3,4,5,6,7,8,9

Units/representation
j=1,2,3,7,8,9 Meters
j=4,5 Dimensionless
j=6 Degrees
all Single floats

Treatment/usage
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j) lists the mean value for adjustments for a subswath at subswath level *Iswl.*
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 1) =ECEF x shift
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 2) =ECEF y shift
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 3) =ECEF z shift
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 4) = ECEF x-direction cosine of rotation axis
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 5) = ECEF y-direction cosine of rotation axis
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 6) = angle of rotation about about rotation axis.
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 7) = ECEF x coordinate of the "pin" point on the rotation axis.
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 8) = ECEF y coordinate of the "pin" point on the rotation axis.
PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 9) = ECEF z coordinate of the "pin" point on the rotation axis.
The x- direction cosine measures the angle from the ECEF frame x-axis to the rotation axis determined in the registration process yielding the adjustment for the subswath described by (*Iswl*, isw). Similarly, the y- direction cosine measures the angle from the ECEF frame y-axis to the rotation axis determined in the registration process yielding the adjustment for the subswath described by (*Iswl*, isw). The nature of the input point blocks used in the registration that determines the adjustments provided in PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j) are found in PF.Registn.SubswathDat(iswl,3).
A 3 x 3 rotation matrix can be formed from the axis, angle data in elements 4,5, and 6.
c = cos(θ); s = sin(θ); C = 1-c; x= PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 4)
y= PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 5);
θ= PF.Adjustmts.Mean.Subswath.*Iswl*(isw, 6)
$z=\mathrm{sqrt}(1-x^2-y^2)$
xs = x*s; ys = y*s; zs = z*s (* denotes multiplication)
xC = x*C; yC = y*C; zC = z*C
xyC = x*yC; yzC = y*zC; zxC = z*xC

$$R_{ax,ang} = \begin{bmatrix} x*xC+c & xyC-zs & zxC+ys \\ xyC+zs & y*yC+c & yzC-xs \\ zxC-ys & yzC+xs & z*zC+c \end{bmatrix}$$

The "pin" point, whose ECEF coordinates are in the j=7,8,9 elements for each set of subswath adjustment data, may be treated as the origin for a vector whose direction is given via the j=4, 5 elements of PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j).

The model for documenting adjustments is that a down stream application/user would create a net correction to be applied (or analyzed/assessed if it has already been applied) by accumulating adjustments by walking a tree of adjustments. For example, we pick a point from the point cloud and noting its position and time of collection [from ES.Point.PulseNum.*TileNum*(ipt, j)], we determine a sequence of a scan through subswaths to which it has been assigned all the way to the final subswath layer.

> {SC.Pulse.Scan.Num.*TileNum*(ip, j) maps a pulse into a scan number. PF.Subswath.ScansList.*iswl*(isw,it,j) maps scans into subswaths by layer. A point maps to a single time (IPP) and scan. It could map to 1,2,3 or 4 subswaths at any layer below the full swath.}

We then add the mean correction and linearly interpolated local demeaned adjustment for each layer into a shift accumulator. Moving up through the layers we find corrections based on successively larger regions of adjustment tracking. If we are looking to accumulate a correction to apply to a candidate point, only those adjustments marked as "not applied" (see PF.Registn.*) would be included in the sum.

The shift accumulation process just described will yield correct total shifts for any point despite the assignment of some points in overlap regions of processing subswaths to more than one subswath in any subswath layer contributing to the final swath. This may seem surprising since a point could receive a different shift when being assigned to different subswaths in one level. However:

a. Subswaths are registerable because common points in each are subject to different shifts in the process of building them up into each subswath in which they are included. This means that the coordinate labels assigned to a single point can be different for each subswath containing it. Because we are talking of a single ECEF true point position, this means that each time we aggregate a point into a subswath, we are assigning a subswath specifc frame to it and naming its coordinates accordingly.
b. Registration of subswaths at one layer to form one subswath at the next layer renames the multiple coordinate labels for a single point to a single coordinate

label. That means a shift is found for each constituent subswath at the previous layer that just remedies the previous shift "inconsistencies".

c. If we save the shifts to get to a cumulative shift at each step, it matters not which subswaths we assign a point to as long as the shift list is consistent so that the shifts are those of the subswaths to which we have been assigning the point. The sums for each of the subswath paths for a given point would then give the same final point shift from the original frame level direction-range-sensor position/attitude point assignment to ECEF.

Conclusion: We could get the right final shift if we only save the shifts for the "lowest" subswath number that a point resides in at each layer. This puts overlap points into the same strategy as points in non-overlap parts of the subswath without distinction in the treatment.

Default if not populated

PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j) is assumed to have all elements be unknown if they are not explicitly given a value and PD.Adjustments.Diffs.Subswath.*Iswl* are also not populated. However, if PD.Adjustments.Diffs.Subswath.*Iswl* are populated and PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j) is not, an error condition should be indicated. This forces explicit population of zero mean shift data in the rare occasion that the differential shifts are natively zero mean without explicitly demeaning the shift data.

Governing quality metrics: The pin point shall be chosen so that the mean shift over the affected subswath due to the adjustment rotation is no larger than the larger of 1 percent of the length of the mean adjustment shift vector or the hard bound amount indicated in PF.Registn.SubswathDat(4) .

See Also: PD.HiIndx.Overlap(4) PF.Registn.SubswathDat(3)

20.3.5 PF.Adjustmts.Diffs.Positions.Scan(is,i,j) [L2,3,4,5]

Brief Definition: PF.Adjustmts.Diffs.Positions.Scan(is, i, j) provides the earth frame spatial positions to which demeaned shifts for an aligning transformation at the scan level in the creation of subswaths are attributed.

indices: Index, is, ranges from PD.LoIndx.Scan(3) to PD.HiIndx.Scan(3), that is, over the number of scans in the dataset.
Index, i, ranges from PD.LoIndx.Scan(6) to PD.HiIndx.Scan(6), that is, over the number of shift adjustments documented for scans. This range of "i" is the same for all scans in a dataset.
Index, j= 1,2,3 {or 0,1,2 depending on the implementation language} covering the XYZ components of positions

Units/representation: Centimeters, Int32

Treatment/usage: If more than a single mean shift has been determined for geometrically correcting the positions of points at the scan level, the demeaned shift values

are listed in PF.Adjustmts.Diffs.Positions.Scan(is, i, j). Those shifts are to be interpreted as belonging to the isth scan at ECEF positions listed here in PF.Adjustmts.Diffs.Positions.Scan(is, i, j). The ith position for which an adjusting shift is supplied is provided as follows:

PF.Adjustmts.Diffs.Positions.Scan(is, i, 1) = ECEF X coordinate
PF.Adjustmts.Diffs.Positions.Scan(is, i, 2) = ECEF Y coordinate
PF.Adjustmts.Diffs.Positions.Scan(is, i, 3) = ECEF Z coordinate

The integer valued x,y,z coordinates in meters/100 (centimeters) provided for the isth scan are directly populated **without** use of a linear transformation as is applied to the point cloud entries.

The geometrical adjustment for the isth scan may have been derived from intra- or inter-swath registration operations as documented in PF.Registn.ScanDat(3)

Governing quality metrics: The positions indicated by PF.Adjustmts.Diffs.Positions.Scan(is, i, j) must lie in or close to the boundary of the volume of the lidar data returns for the isth scan.

See Also: PD.HiIndx.Scan(3) PF.Registn.ScanDat(3) PD.HiIndx.Scan(j)

20.3.6 PF.Adjustmts.Diffs.Positions.SubSwath.*Iswl*(isw,i,j) [L2,3,4,5]

Brief Definition: PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j) provides the earth frame spatial positions to which demeaned shifts for an aligning transformation at thew subswath level in the creation of a larger aggregation of points are attributed.

indices: Computed substring "*Iswl*" acts somewhat like a subscript and ranges over the subswath layers, of which there are PF.Subswath.Nlayers. By making this a substring, we create an array for each subswath layer so that we can provide different length lists of adjustments in each subswath layer. The *Iswl* substring will be a character representation of the numerical value of the subswath layer being treated by the array, PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). In this discussion, 'iswl' is the integer corresponding to the value of the character string "*Iswl*".

Index, isw, ranges over the number of subswaths in the subswath layer indicated by the value of substring *Iswl*, that is, effectively from 1 to PF.Subswath.NumSubswaths(iswl).

Index, i, ranges from PD.LoIndx.SubSwAdj(iswl) to PD.HiIndx.SubSwAdj(iswl), that is, over the number of shift adjustments

	documented for subswaths at subswath layer iswl. This range of "i" is the same for all subswaths in a subswath layer for a dataset. Index, j= 1,2,3 {or 0,1,2 depending on the implementation language} covering the XYZ components of positions
Units/representation	Centimeters, Int32
Treatment/usage	If more than a single mean shift has been determined for geometrically correcting the positions of points at the subswath level, the demeaned shift values are listed in PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). Those shifts are to be interpreted as belonging to the iswth subswath at subswath layer "iswl" at the ECEF positions listed here in PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). The ith position for which an adjusting shift is supplied is provided as follows: PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, 1) = ECEF X coordinate PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, 2) = ECEF Y coordinate PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, 3) = ECEF Z coordinate The integer valued x,y,z coordinates in meters/100 (centimeters) provided for the subswath indicated by isw and *Iswl* are directly populated **without** use of a linear transformation as is applied to the point cloud entries. The geometrical adjustment for the isth scan may have been derived from intra- or inter-swath registration operations as documented in PF.Registn.SubswathDat(3)
Governing quality metrics	The positions indicated by PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw,i,j) must lie in or close to the boundary of the volume of the lidar data returns for the subswath indicated by isw and *Iswl*.
See Also	PD.HiIndx.SubSwAdj(iswl) PF.Registn.SubswathDat(3)

20.3.7 PF.Adjustmts.Diffs.Shifts.Scan(is,i,j) [L2,3,4,5]

Brief Definition	PF.Adjustmts.Diffs.Shifts.Scan(is,i, j) provides the shifts for the aligning transformation at the scan level in the creation of subswaths.
indices	Index, is, ranges from PD.LoIndx.Scan(3) to PD.HiIndx.Scan(3), that is, over the number of scans in the dataset. Index, i, ranges from PD.LoIndx.Scan(6) to PD.HiIndx.Scan(6), that is, over the number of shift adjustments documented for scans. This range of "i" is the same for all scans in a dataset. Index, j= 1,2,3
Units/representation	Meters, Single float

Treatment/usage

For point cloud aggregations larger than the single pulse level, registration operations have a reasonable likelihood of detecting spatially variable misalignments. Thus in addition to the mean adjustment for the points in a single scan, we need to document a set of locally distinct adjustments. This is accomplished by providing a list of positions at which the values of shifts in the ECEF frame are attached. The positions in a list are provided to allow for nonuniform coverage so that spatially inhomogeneous geometry errors can be presented, but the data could be provided on a regular grid when appropriate. These positions for the scan level adjustments are given by PF.Adjustmts.Diffs.Positions.Scan(is, i, j). The shifts are provided here as follows:

PF.Adjustmts.Diffs.Shifts.Scan(is, i, 1) = ECEF demeaned x shift
PF.Adjustmts.Diffs.Shifts.Scan(is, i, 2) = ECEF demeaned y shift
PF.Adjustmts.Diffs.Shifts.Scan(is, i, 3) = ECEF demeaned z shift

Note that these shifts are stated to be demeaned. That is so that they can be applied as additional corrections to point positions after applying the shifts documented in PF.Adjustmts.Mean.Scan(is, j).

The shifts are presented on a field of positions sampled sufficiently densely that tri-linear interpolation will be sufficient to meet the system accuracy and geometric distortion requirements of the sensor system providing the lidar dataset.

While it is very likely that adjustments at the scan level will be applied in late level-2 processing stages and higher product levels, this CMMD supports documenting the applied adjustments or saving them for application later in exploitation.

The model for documenting adjustments is that a down stream application/user would create a net correction to be applied (or analyzed/assessed if it has already been applied) by accumulating adjustments by walking a tree of adjustments. For example, we pick a point from the point cloud and noting its position and time of collection [from ES.Point.PulseNum.*TileNum*(ipt, j)], we determine a sequence of a scan through subswaths to which it has been assigned all the way to the final subswath layer.

> {SC.Pulse.Scan.Num.*TileNum*(ip, j) maps a pulse into a scan number. PF.Subswath.ScansList.*iswl*(isw,it,j) maps scans into subswaths by layer. A point maps to a single time (IPP) and scan. It could map to 1,2,3 or 4 subswaths at any layer below the full swath.}

We then add the mean correction and linearly interpolated local demeaned

adjustment for each layer into a shift accumulator. Moving up through the layers we find corrections based on successively larger regions of adjustment tracking. At the bottom of the adjustment stack, at the smallest scale at which spatial variability of adjustments is documented is the scan level whose shifts are found in PF.Adjustmts.Diffs.Shifts.Scan(is,i, j). If we are looking to accumulate a correction to apply to a candidate point, only those adjustments marked as "not applied" (see PF.Registn.*) would be included in the sum.

The shift accumulation process just described will yield correct total shifts for any point despite the assignment of some points in overlap regions of processing subswaths to more than one subswath in any subswath layer contributing to the final swath. This may seem surprising since a point could receive a different shift when being assigned to different subswaths in one level. However:

d. Subswaths are registerable because common points in each are subject to different shifts in the process of building them up into each subswath in which they are included. This means that the coordinate labels assigned to a single point can be different for each subswath containing it. Because we are talking of a single ECEF true point position, this means that each time we aggregate a point into a subswath, we are assigning a subswath specifc frame to it and naming its coordinates accordingly.

e. Registration of subswaths at one layer to form one subswath at the next layer renames the multiple coordinate labels for a single point to a single coordinate label. That means a shift is found for each constituent subswath at the previous layer that just remedies the previous shift "inconsistencies".

f. If we save the shifts to get to a cumulative shift at each step, it matters not which subswaths we assign a point to as long as the shift list is consistent so that the shifts are those of the subswaths to which we have been assigning the point. The sums for each of the subswath paths for a given point would then give the same final point shift from the original frame level direction-range-sensor position/attitude point assignment to ECEF.

Conclusion: We could get the right final shift if we only save the shifts for the "lowest" subswath number that a point resides in at each layer. This puts overlap points into the same strategy as points in non-overlap parts of the subswath without distinction in the treatment. This means that PF.Adjustmts.Diffs.Shifts.Scan(is,i, j) needs no subswath index. The registrati toons executed to build the subswath with the lowest subswath number in the first layer of subswaths determines the shifts that will be stored for a particular scan.

The spacing of positions for which shifts are provided is chosen to be small enough that neglecting local rotations will not violate the system geometric accuracy requirements for the given dataset. If the shifts are presented in a list of collinear positions, the demeaned shift field should be interpreted as one dimensional. Similarly, if the listed positions are planar to the precision

of the position data, the demeaned shift field should be interpreted as two dimensional.

Default if not populated
PF.Adjustmts.Diffs.Shifts.Scan(is,i, j) is assumed to have all elements be zero if they are not explicitly given a value.

Governing quality metrics

See Also PD.HiIndx.Overlap(3) PF.Registn.ScanDat(3) SC.Pulse.Scan.Num.*TileNum*(ip,j) PF.Subswath.ScansList.*iswl*(isw,it,j)

20.3.8 PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i,j) [L2,3,4,5]

Brief Definition PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, j) provides the aligning coordinate transformation at subswath layerl *Iswl* in the creation of larger subswaths or the final swath.

indices Computed substring "*Iswl*" acts somewhat like a subscript and ranges over the subswath layers, of which there are PF.Subswath.Nlayers. By making this a substring, we create an array for each subswath layer so that we can provide different length lists of adjustments in each subswath layer. The *Iswl* substring will be a character representation of the numerical value of the subswath layer being treated by the array, PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). In this discussion, 'iswl' is the integer corresponding to the value of the character string "*Iswl*".

Index, isw, ranges over the number of subswaths in the subswath layer indicated by the value of substring *Iswl*, that is, effectively from 1 to PF.Subswath.NumSubswaths(iswl).

Index, i, ranges from PD.LoIndx.SubSwAdj(iswl) to PD.HiIndx.SubSwAdj(iswl), that is, over the number of shift adjustments documented for subswaths at subswath layer iswl. This range of "i" is the same for all subswaths in a subswath layer for a dataset.
Index, j= 1,2,3 {or 0,1,2 depending on the implementation language} covering the XYZ components of shifts.

Units/representation Meters, Float

Treatment/usage For point cloud aggregations larger than the single pulse level, registration operations have a reasonable likelihood of detecting spatially variable misalignments. Thus in addition to the mean adjustment for the points in a single subswath, we need to document a set of locally distinct adjustments. This is accomplished by providing a list of positions at which the values of shifts in the ECEF frame are attached. The positions in a list are provided to

allow for nonuniform coverage so that spatially inhomogeneous geometry errors can be presented, but the data could be provided on a regular grid when appropriate. These positions for the subswath *Iswl* level adjustments are given by PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, j). The shifts are provided here as follows:
PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, 1) = ECEF demeaned x shift
PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, 2) = ECEF demeaned y shift
PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, 3) = ECEF demeaned z shift

Note that these shifts are stated to be demeaned. That is so that they can be applied as additional corrections to point positions after applying the shifts documented in PF.Adjustmts.Mean.Subswath.*Iswl*(isw,i, j).

The shifts are presented on a field of positions sampled sufficiently densely that tri-linear interpolation will be sufficient to meet the system accuracy and geometric distortion requirements of the sensor system providing the lidar dataset.

While it is possible that adjustments for any subswath layer will be applied in late level-2 processing stages and higher product levels, this CMMD supports documenting the applied adjustments or saving them for application later in exploitation.

For an explanation of determining the total adjustment for any point position and how PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, 1) contributes to the final shifts to be applied to a point, see PF.Adjustmts.Diffs.Shifts.Scan(is,i, j).

The spacing of positions for which shifts are provided is chosen to be small enough that neglecting local rotations will not violate the system geometric accuracy requirements for the given dataset. If the shifts are presented in a list of collinear positions, the demeaned shift field should be interpreted as one dimensional. Similarly, if the listed positions are planar to the precision of the position data, the demeaned shift field should be interpreted as two dimensional.

Default if not populated
PF.Adjustmts.Diffs.Shifts.SubSwath.*Iswl*(isw,i, j) is assumed to have all elements be zero if they are not explicitly given a value.

Governing quality metrics

See Also

PD.HiIndx.Overlap(4) PF.Registn.SubswathDat(3)
PF.Adjustmts.Diffs.Shifts.Scan(is,i, j) PD.LoHiIndx.SubSwathAdj(iswl)

PF.Channlzn.*

PF.Channlzn.* parameters describe the lidar processing history and intent for channels represented in level-2,3,4 data sets

20.3.9 PF.Channlzn.Descr.*m*(j) [L2,3,4,5]

Brief Definition

PF.Channlzn.Descr.*m*(j) provides a description for each channel of data identified in a lidar data set.

indices

This is a reference to the jth channel of the mth channelization.
The evaluated tag substring, *m*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a separately addressable block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The character evaluation of *m* is a numeric symbol and its integer conversion ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is, over the number of channelizations {intermediate processing stages for which the PF.Channlzn.Descr.*m*(j) data array has been populated (includes both output/stored data sets and what may be deemed as internal transient versions of the data within the processor).} The range of *m* will increase by one each time a new channelization[147] is created. The convention is that *m*=0 refers to the enterprise level-1 product data set (i.e. the product that is assured to have only the collection channels), *m*=1 refers to the first post L-1 stage, *m*=2 refers to the second post L-1 stage, etc. There is a degree of arbitrariness to defining the interim channelizations, so there is some discretion left to the processors in how to populate any channelizations that they create but do not use in their output WB data. But, once a channelization description has been added to a product chain by populating PF.Channlzn.Descr.*m*(j), it should be inherited with the rest of the channelization descriptions in subsequent data set generation. Only the current data set will have an assigned channelization, that being associated with the highest value for *m* in the data set. The precursor output data sets with channelizations will be identified in PF.Recon.PrevProdNames(mm,k), but only when one is examining the PF.Channlzn.Descr.*m*(j), populated in each of them will their as-processed channelization be certain whenever the

[147] To be clear, a channelization is the collection of all the final channels of a processing stage. The value of *m* is not to be incremented until the complete set of new channels is considered. This avoids seeing a serial algorithm make higher m counts by incrementing for each new output channel on the way to creating a new channelization.

number of channelizations exceeds the number of previous products listed in PF.Recon.PrevProdNames(mm,k).

Index, j, ranges from from PD.LoIndx.Channlzn.ChRg(m) to PD.HiIndx.Channlzn.ChRg(m), where m is the integer translation of *m*. Only values of *m* from (PD.LoIndx.Channlzn.Rg_m+1) to PD.HiIndx.Channlzn.Rg_m should appear in PF.Channlzn.Descr.*m*(j). That is, j ranges over the as-processed channels present the level-2 or higher channelizations of the processing chain ending in the current product. To be clear, we have no range of j for m= PD.LoIndx.Channlzn.Rg_m. That channelization index refers to the collection channels and is addressed in SC.Channlzn.Descr(j,k). To describe the channels after processing that affects the channelization, j for *m*= (PD.LoIndx.Channlzn.Rg_m +1) ranges over the number of channels in the first post L-1 stage, j for *m*= (PD.LoIndx.Channlzn.Rg_m +2) ranges over the number of channels in the second post L-1 stage, etc. The number of channels in a channelization can vary across channelizations, so we structure the channelization descriptions so that the range of j for each channelization is separately specifiable. This is why we put *m* into the string name and did not build a formal array. The channelization for the current data set is given by the PF.Channlzn.Descr.*m*(j) with the highest value of *m.* That value of *m* is found in PD.HiIndx.Channlzn.Rg_m.

Units/representation Dimensionless, Character(80)

Treatment/usage Unlike the collection channels defined in SC.Channlzn.Descr(j), which describe the sensor's diversity in spectral, polarization or field of view domains, PF.Channlzn.Descr.*m*(j) is used to describe channels in level-2 and above products. While, it is possible that the original collection channels have been preserved in these higher level products, it is also permitted that the channels have been combined in some fashion to create new, and perhaps a fewer number of channels.
PF.Channlzn.Descr.*m*(j) provides text description to elaborate information provided in PF.Channlzn.Hist.ChTrace.*m*(j,k), PF.Channlzn.Hist.Comb.Code.*m*(j), PF.Channlzn.Hist.Comb.Func.*m*(k), and PF.Channlzn.Hist.Comb.Alg.*m*(k) when populated.
Text guidance:

- In references to scattering matrix components, the descriptions should use the convention that the first symbol is the transmit and the second is the receive power, as in (VH) meaning vertical transmit, horizontal receive.
- In references to spectral channels, λ_p should represent the wavelength

of the peak of a spectral band.

- In making reference to channels, Ch_i_{mn} should represent an input channel to a combination procedure as being the nth channel in the mth channelization, not just the current product output channelization being described, and Ch_o_{mn} should represent an output channel from a combination procedure as the nth channel in the mth channelization.

Additional descriptive conventions as need arises will be added here.
For rules on synchronizing the m dimension across data sets and PF.Channlzn.*, see discussion of channelization history in 10.1.3.1.

If PF.Channlzn.Descr.*m*(j) is not populated for a multichannel lidar data set at L2 or higher, then the channels are to be taken as described by SC.Channlzn.Descr(j,k). Note that when we go to an enterprise L3 product we may[148] process the points from multiple channels into multiple channel points. That does not, in itself, imply a change in the channelization.

Governing quality metrics

See Also

SC.Channlzn.Descr(j) PF.Channlzn.Hist.ChTrace(j,k), PF.Channlzn.Hist.Comb.Code(j), PD.HiIndx.Channlzn.Rg_m PF.Channlzn.Hist.Comb.Func(k), PF.Channlzn.Hist.Comb.Alg(k) PF.Recon.PrevProdNames(mm,k)

20.3.10 PF.Channlzn.Disp.*m*(j) [L2,3,4,5]

Brief Definition

PF.Channlzn.Disp.*m*(j) documents the intended use of channels in a lidar data set

indices

This is a reference to the jth channel of the mth channelization.
The evaluated tag substring, *m*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a separately addressable block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The character evaluation of *m* is a numeric symbol and its integer conversion ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that

[148] Only the case where the channels were collected as spatially diverse might we preserve channel unique points in a channel combination process performed in going to a level-3. Typically, the channels would be combined down to a single channel with all of the points in this case. They may be sorted into tiles by the collection aspect angles, but channel identity would be intentionally lost.

is, over the number of channelizations {intermediate processing stages for which the PF.Channlzn. Disp.*m*(j) data array has been populated (includes both output/stored data sets and what may be deemed as internal transient versions of the data within the processor).} The range of *m* will increase by one each time a new channelization[147] is created. The convention is that *m*=0 refers to the enterprise level-1 product data set (i.e. the product that is assured to have only the collection channels), *m*=1 refers to the first post L-1 stage, *m*=2 refers to the second post L-1 stage, etc. There is a degree of arbitrariness to defining the interim channelizations, so there is some discretion left to the processors in how to populate any channelizations that they create but do not use in their output WB data. But, once a channelization description has been added to a product chain by populating PF.Channlzn. Disp.*m*(j), it should be inherited with the rest of the channelization descriptions in subsequent data set generation. Only the current data set will have an assigned channelization, that being associated with the highest value for *m* in the data set. The precursor output data sets with channelizations will be identified in PF.Recon.PrevProdNames(mm,k), but only when one is examining the PF.Channlzn. Disp.*m*(j), populated in each of them will their as-processed channelization be certain whenever the number of channelizations exceeds the number of previous products listed in PF.Recon.PrevProdNames(mm,k).

Index, j, ranges from from PD.LoIndx.Channlzn.ChRg(m) to PD.HiIndx.Channlzn.ChRg(m), where m is the integer translation of *m*. Only values of *m* from (PD.LoIndx.Channlzn.Rg_m+1) to PD.HiIndx.Channlzn.Rg_m should appear in PF.Channlzn.Disp.*m*(j). That is, j ranges over the as-processed channels present the level-2 or higher channelizations of the processing chain ending in the current product. To be clear, we have no range of j for m= PD.LoIndx.Channlzn.Rg_m. That channelization index refers to the collection channels and is addressed in SC.Channlzn. Disp(j,k). To describe the channels after processing that affects the channelization, j for *m*= (PD.LoIndx.Channlzn.Rg_m +1) ranges over the number of channels in the first post L-1 stage, j for *m*= (PD.LoIndx.Channlzn.Rg_m +2) ranges over the number of channels in the second post L-1 stage, etc. The number of channels in a channelization can vary across channelizations, so we structure the channelization descriptions so that the range of j for each channelization is separately specifiable. This is why we put *m* into the string name and did not build a formal array. The channelization for the current data set is given by the PF.Channlzn.Descr.*m*(j) with the highest value of *m.* That value of *m* is found in PD.HiIndx.Channlzn.Rg_m.

Units/representation Dimensionless, UInt8

Treatment/usage PF.Channlzn.Disp.*m*(j) supplies a code with values **1** through **12** described in the table below:

The disposition intent for channels in the current data set is a combination of whether or not the channelization is intended to be further processed (intermediate) or is intended to be used as is (Directly exploitable), the fusion space intended when the channels are intermediate (spectral/polarimetric, spatial, temporal) and whether the channel is dominantly signal (column 2), dominantly clutter or noise (column3), or has a significant contribution from both signal and clutter/noise (column 4).

To illustrate, a value of 4 indicates that a channel is intended for a later spectral or polar fusion step and is dominated by clutter or noise while a value of 1 indicates that the channel is intended for later spectral or polar fusion but is signal dominated. The value, 12, is present for logical completeness but is not likely to be a widely pursued strategy [because it is a nearly even mix of "signal" and clutter] while 10 is a very likely disposition for a clean channel ready for easy direct analyst interpretation.

	signal	**Clutter/ noise**	**Signal and clutter**
intermediate	**1.** Intended for spectral or polar fusion step **2.** Intended for spatial fusion step **3.** Intended for temporal fusion step	**4.** Intended for spectral or polar fusion step **5.** Intended for spatial fusion step **6.** Intended for temporal fusion step	**7.** Intended for spectral or polar fusion step **8**. Intended for spatial fusion step **9**. Intended for temporal fusion step
Directly exploitable	**10**	**11**	**12**

For rules on synchronizing the mth channelization across data sets and PF.Channlzn.*, see discussion of channelization history in 10.1.3.1.

Governing quality metrics

See Also

20.3.11 PF.Channlzn.Hist.ChTrace.*m*(j,k) [L2,3,4,5]

Brief Definition PF.Channlzn.Hist.ChTrace.*m*(j,k) indicates the parent channels for each channel in the mth channelization appearing in an enterprise L-2 or higher lidar data set.

indices The evaluated tag substring, *m*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a

vector of values in a separately addressable block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The character evaluation of *m* is a numeric symbol and its integer conversion ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is, over the number of channelizations {intermediate processing stages for which the PF.Channlzn.Hist.ChTrace.*m*(j) data array has been populated (includes both output/stored data sets and what may be deemed as internal transient versions of the data within the processor).} The range of *m* will increase by one each time a new channelization[147] is created. The convention is that *m*=0 refers to the enterprise level-1 product data set (i.e. the product that is assured to have only the collection channels), *m*=1 refers to the first post L-1 stage, *m*=2 refers to the second post L-1 stage, etc. There is a degree of arbitrariness to defining the interim channelizations, so there is some discretion left to the processors in how to populate any channelizations that they create but do not use in their output WB data. But, once a channelization has been added to a product chain by populating PF.Channlzn.Hist.ChTrace.*m*(j), it should be inherited with the rest of the channelization descriptions in subsequent data set generation. Only the current data set will have an assigned channelization, that being associated with the highest value for *m* in the data set. The precursor output data sets with channelizations will be identified in PF.Recon.PrevProdNames(mm,k), but only when one is examining the PF.Channlzn.Descr.*m*(j), populated in each of them will their as-processed channelization be certain whenever the number of channelizations exceeds the number of previous products listed in PF.Recon.PrevProdNames(mm,k).

Index, j, ranges from PD.LoIndx.Channlzn.ChRg(*m*-1) to PD.HiIndx.Channlzn.ChRg(*m*-1), that is, over the parent channels in the $(m-1)^{th}$ channelization that act as inputs for creating the mth channelization.
Index, k, ranges from PD.LoIndx.Channlzn.ChRg(*m*) to PD.HiIndx.Channlzn.ChRg(*m*), that is, over the as-processed channels present in the mth channelization.

Units/representation
Dimensionless, Float

Treatment/usage
Channels can be inherited as-is from parent files, or the channels in a parent file can be combined to create new, and probably fewer, channels in the daughter product. PF.Channlzn.Hist.ChTrace.*m*(j,k) acts as a table indicating which channels of the parent data set contributed to which channels of the current data set.

Each row of PF.Channlzn.Hist.ChTrace.*m*(j,k) is dedicated to a parent channel and each column in that row will have a value corresponding to a non- zero weight if that parent contributes to the channel in the current data set corresponding to the column number, and will have a value of 0.0E00 if that parent channel has zero weight in the forming the channel in the current data set. When PF.Channlzn.Hist.Comb.Code.*m*(j) is set to 3 or 4, the values in PF.Channlzn.Hist.ChTrace.*m*(j,k) correspond to combination weights. When PF.Channlzn.Hist.Comb.Code.*m*(j) is set to 5 or 6, values of 1.0E00 and 0.0E00 should be used to indicate whether (1.0E00) or not (0.0E00) the available parent file channel contributes in the more complicated functions or algorithms indicated by *Comb.Func.*m*(k) and *Comb.Alg.*m*(k).
In a channelization (indexed by *m*), each column of PF.Channlzn.Hist.ChTrace.*m*(j,k) is dedicated to a channel in the current data set.

PF.Channlzn.Hist.ChTrace.*m*(j,k)

	***m*th ch 1**	**…**	***m*th ch K**
(*m*-1)th ch 1	.ChTrace.*m*(1,1)		.ChTrace.*m*(1,K)
(*m*-1)th ch 2			
…			
(*m*-1)th ch J	.ChTrace.*m*(J,1)		.ChTrace.*m*(J,K)

In the above table, J=PD.HiIndx.Channlzn.ChRg(*m*-1) and K=PD.HiIndx.Channlzn.ChRg(*m*).

PF.Channlzn.Hist.ChTrace.*m*(j,k)≠0 for table entries when the parent channel, j=1 to J, contributes to product channel, k=1 to K. If the process to combine the channels is one of linear superposition[149], PF.Channlzn.Hist.ChTrace.*m*(j,k) may be set to be the weights. PF.Channlzn.Hist.ChTrace.*m*(j,k)=0 otherwise. {i.e. when a channel is zero weighted in the combination process}
For rules on synchronizing the *m* channelizations across data sets and PF.Channlzn.*, see discussion of channelization history in 10.1.3.1

Governing quality metrics

See Also

PD.HiIndx.Channlzn.ChRg(*m*) PF.Channlzn.Hist.Comb.Code.m(j)
PF.Channlzn.Hist.Comb.Func.*m*(k) PF.Channlzn.Hist.Comb.Alg.*m*(k)

20.3.12 PF.Channlzn.Hist.Comb.Code.*m*(j) [L2,3,4,5]

Brief Definition

PF.Channlzn.Hist.Comb.Code.*m*(j) documents the combination methodology

[149] Or one in which an apriori multiple of a channel is the input to the channel combination process

at top level for each precursor and current channelization.

indices

This is a reference to the jth channel of the mth channelization.
The evaluated tag substring, *m*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a separately addressable block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The character evaluation of *m* is a numeric symbol and its integer conversion ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is, over the number of channelizations {intermediate processing stages for which the PF.Channlzn.Hist.Comb.Code.*m*(j) data array has been populated (includes both output/stored data sets and what may be deemed as internal transient versions of the data within the processor).} The range of *m* will increase by one each time a new channelization[147] is created. The convention is that *m*=0 refers to the enterprise level-1 product data set (i.e. the product that is assured to have only the collection channels), *m*=1 refers to the first post L-1 stage, *m*=2 refers to the second post L-1 stage, etc. There is a degree of arbitrariness to defining the interim channelizations, so there is some discretion left to the processors in how to populate any channelizations that they create but do not use in their output WB data. But, once a channelization description has been added to a product chain by populating PF.Channlzn.Hist.Comb.Code.*m*(j), it should be inherited with the rest of the channelization descriptions in subsequent data set generation. Only the current data set will have an assigned channelization, that being associated with the highest value for *m* in the data set. The precursor output data sets with channelizations will be identified in PF.Recon.PrevProdNames(mm,k), but only when one is examining the PF.Channlzn.Descr.*m*(j), populated in each of them will their as-processed channelization be certain whenever the number of channelizations exceeds the number of previous products listed in PF.Recon.PrevProdNames(mm,k).

Index, j, ranges from from PD.LoIndx.Channlzn.ChRg(m) to PD.HiIndx.Channlzn.ChRg(m), where m is the integer translation of *m*. Only values of *m* from (PD.LoIndx.Channlzn.Rg_m+1) to PD.HiIndx.Channlzn.Rg_m should appear in PF.Channlzn.Descr.*m*(j). That is, j ranges over the as-processed channels present the level-2 or higher channelizations of the processing chain ending in the current product. To be clear, we have no range of j for m= PD.LoIndx.Channlzn.Rg_m. That channelization index refers to the collection channels and is addressed in SC.Channlzn.Descr(j,k). To describe the channels after processing that

affects the channelization, j for *m*= (PD.LoIndx.Channlzn.Rg_m +1) ranges over the number of channels in the first post L-1 stage, j for *m*= (PD.LoIndx.Channlzn.Rg_m +2) ranges over the number of channels in the second post L-1 stage, etc. The number of channels in a channelization can vary across channelizations, so we structure the channelization descriptions so that the range of j for each channelization is separately specifiable. This is why we put *m* into the string name and did not build a formal array. The channelization for the current data set is given by the PF.Channlzn.Descr.*m*(j) with the highest value of *m*. That value of *m* is found in PD.HiIndx.Channlzn.Rg_m.

Units/representation
Dimensionless, UInt8

Treatment/usage
While level-1 data sets are limited to as-collected channels, it is possible that the collected channels may be combined to form new channels, likely fewer in number than originally collected, when creating level-2 and level-3 products. PF.Channlzn.Hist.Comb.Code.*m*(j) supplies codes that are consistent with descriptions that may be populated into PF.Channlzn.Descr.*m*(j). These codes characterize how the mth channelization was created.

codes:

PF.Channlzn.Hist.Comb.Code.*m*(j) =0.
Channels are as collected and retain original identity

PF.Channlzn.Hist.Comb.Code.*m*(j) =1.
Spatial juxtaposition-- Point file merge without adjustment

PF.Channlzn.Hist.Comb.Code.*m*(j) =2.
Spatial juxtaposition-- Point file merge with adjustment (registration, alignment)

PF.Channlzn.Hist.Comb.Code.*m*(j) =3.
Linear superposition with weights W_i. [weights given in PF.Channlzn.Hist.ChTrace(j,k,m)]

PF.Channlzn.Hist.Comb.Code.*m*(j) =4.
RSS (possibly weighted) [weights given in PF.Channlzn.Hist.ChTrace(j,k,m)]

PF.Channlzn.Hist.Comb.Code.*m*(j) =5.
More general function expressible by a simple equation Cn=f(c1,c2,c3,…) [see PF.Channlzn.Hist.Comb.Func(k,m)]

PF.Channlzn.Hist.Comb.Code.*m*(j) =6.
A complicated algorithm with conditional, likely data dependent steps for combination. Such an algorithm needs to have a registered name [could be listed in the CMMD] which can be included as metadata to support documentation of how the data has been processed. [see PF.Channlzn.Hist.Comb.Alg.*m*(k)]

…
The process described by the above codes is applied to precursor file channels and assigned to the mth channelization per PF.Channlzn.Hist.ChTrace.*m*(j,k). For rules on synchronizing the mth channelization across data sets and PF.Channlzn.*, see discussion of channelization history in 10.1.3.1.

Governing quality metrics

See Also

PF.Channlzn.Descr.*m*(j) PF.Channlzn.Hist.ChTrace.*m*(j,k)
PF.Channlzn.Hist.Comb.Func.*m*(k) PF.Channlzn.Hist.Comb.Alg.*m*(k)

20.3.13 PF.Channlzn.Hist.Comb.Func.*m*(j) [L2,3,4,5]

Brief Definition

PF.Channlzn.Hist.Comb.Func.*m*(j) provides second level detail for channels created using simple equations other than RSS and Linear combination.

indices

This is a reference to the jth channel of the mth channelization.
The evaluated tag substring, *m*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a separately addressable block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The character evaluation of *m* is a numeric symbol and its integer conversion ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is, over the number of channelizations {intermediate processing stages for which the PF.Channlzn.Hist.Comb.Func.*m*(j) data array has been populated (includes both output/stored data sets and what may be deemed as internal transient versions of the data within the processor).} The range of *m* will increase by one each time a new channelization[147] is created. The convention is that *m*=0 refers to the enterprise level-1 product data set (i.e. the product that is assured to have only the collection channels), *m*=1 refers to the first post L-1 stage, *m*=2 refers to the second post L-1 stage, etc. There is a degree of arbitrariness to defining the interim channelizations, so there is some discretion left to the processors in how to populate any channelizations that they create but do not use in their output WB data. But, once a channelization description has been added to a product chain by populating PF.Channlzn.Hist.Comb.Func.*m*(j), it should be inherited with the rest of the channelization descriptions in subsequent data set generation. Only the current data set will have an assigned channelization, that being associated with the highest value for *m* in the data set. The precursor output data sets with channelizations will be identified in PF.Recon.PrevProdNames(mm,k), but only when one is examining the PF.Channlzn.Descr.*m*(j), populated in

each of them will their as-processed channelization be certain whenever the number of channelizations exceeds the number of previous products listed in PF.Recon.PrevProdNames(mm,k).

Index, j, ranges from from PD.LoIndx.Channlzn.ChRg(m) to PD.HiIndx.Channlzn.ChRg(m), where m is the integer translation of *m*. Only values of *m* from (PD.LoIndx.Channlzn.Rg_m+1) to PD.HiIndx.Channlzn.Rg_m should appear in PF.Channlzn.Descr.*m*(j). That is, j ranges over the as-processed channels present the level-2 or higher channelizations of the processing chain ending in the current product. To be clear, we have no range of j for m= PD.LoIndx.Channlzn.Rg_m. That channelization index refers to the collection channels and is addressed in SC.Channlzn.Descr(j,k). To describe the channels after processing that affects the channelization, j for *m*= (PD.LoIndx.Channlzn.Rg_m +1) ranges over the number of channels in the first post L-1 stage, j for *m*= (PD.LoIndx.Channlzn.Rg_m +2) ranges over the number of channels in the second post L-1 stage, etc. The number of channels in a channelization can vary across channelizations, so we structure the channelization descriptions so that the range of j for each channelization is separately specifiable. This is why we put *m* into the string name and did not build a formal array. The channelization for the current data set is given by the PF.Channlzn.Descr.*m*(j) with the highest value of *m.* That value of *m* is found in PD.HiIndx.Channlzn.Rg_m.

Units/representation Dimensionless, Character variable length from PD.Wordsize.ChCombFunc

Treatment/usage When the value of PF.Channlzn.Hist.Comb.Code.*m*(j) is set to 5 for a channel, the function used to create a current data set's channel is provided for each channel by an expression in the character string, PF.Channlzn.Hist.Comb.Func.*m*(j). This expression gives the formula for a sample in channel, j, as a function of the input channels, $Chan_j$=f(ch1,ch2,ch3,…) using the syntax of the FORTRAN Language. This can be readily translated to the C language, but as an information character string it is more efficient, mainly owing to the presence of an operator for raising a term to a power. It is understood that references to an input channel assume multiplication by the value of PF.Channlzn.Hist.ChTrace.*m*(j,k) so that weighting process need not be explicitly encoded into the formula in the character strings of PF.Channlzn.Hist.Comb.Func.*m*(j). It is also to be understood that within the character string formula for channel, j, and channelization, *m*, all references to channels (e.g. ch1,ch2,ch3,…) are to channels from channelization (*m*-1).

For rules on synchronizing the *m* channelizations across data sets and

PF.Channlzn.*, see discussion of channelization history in 10.1.3.1

Governing quality metrics

See Also PD.Wordsize.ChCombFunc

20.3.14 PF.Channlzn.Hist.Comb.Alg.*m*(j) [L2,3,4,5]

Brief Definition PF.Channlzn.Hist.Comb.Alg.*m*(j) provides the registered name for complex channel combination algorithms applied in creating the channels within a lidar data set.

indices This is a reference to the jth channel of the mth channelization.
The evaluated tag substring, *m*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a separately addressable block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The character evaluation of *m* is a numeric symbol and its integer conversion ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is, over the number of channelizations {intermediate processing stages for which the PF.Channlzn.Hist.Comb.Alg.*m*(j) data array has been populated (includes both output/stored data sets and what may be deemed as internal transient versions of the data within the processor).} The range of *m* will increase by one each time a new channelization[147] is created. The convention is that *m*=0 refers to the enterprise level-1 product data set (i.e. the product that is assured to have only the collection channels), *m*=1 refers to the first post L-1 stage, *m*=2 refers to the second post L-1 stage, etc. There is a degree of arbitrariness to defining the interim channelizations, so there is some discretion left to the processors in how to populate any channelizations that they create but do not use in their output WB data. But, once a channelization description has been added to a product chain by populating PF.Channlzn.Descr.*m*(j), it should be inherited with the rest of the channelization descriptions in subsequent data set generation. Only the current data set will have an assigned channelization, that being associated with the highest value for *m* in the data set. The precursor output data sets with channelizations will be identified in PF.Recon.PrevProdNames(mm,k), but only when one is examining the PF.Channlzn.Descr.*m*(j), populated in each of them will their as-processed channelization be certain whenever the number of channelizations exceeds the number of previous products listed in PF.Recon.PrevProdNames(mm,k).

Index, j, ranges from from PD.LoIndx.Channlzn.ChRg(m) to PD.HiIndx.Channlzn.ChRg(m), where m is the integer translation of *m*. Only values of *m* from (PD.LoIndx.Channlzn.Rg_m+1) to PD.HiIndx.Channlzn.Rg_m should appear in PF.Channlzn.Hist.Comb.Alg.*m*(j). That is, j ranges over the as-processed channels present the level-2 or higher channelizations of the processing chain ending in the current product. To be clear, we have no range of j for m= PD.LoIndx.Channlzn.Rg_m. That channelization index refers to the collection channels and is addressed in SC.Channlzn.Descr(j,k). To describe the channels after processing that affects the channelization, j for *m*= (PD.LoIndx.Channlzn.Rg_m +1) ranges over the number of channels in the first post L-1 stage, j for *m*= (PD.LoIndx.Channlzn.Rg_m +2) ranges over the number of channels in the second post L-1 stage, etc. The number of channels in a channelization can vary across channelizations, so we structure the channelization descriptions so that the range of j for each channelization is separately specifiable. This is why we put *m* into the string name and did not build a formal array. The channelization for the current data set is given by the PF.Channlzn.Descr.*m*(j) with the highest value of *m*. That value of *m* is found in PD.HiIndx.Channlzn.Rg_m.

Units/representation: Dimensionless Character(30)

Treatment/usage: When PF.Channlzn.Hist.Comb.Code.*m*(j)=6, we can supply a registered character string for the channel combining algorithm[150] used for each channel of the current product. The registry management of the formal algorithm names is TBD, but this paragraph could serve as a listing of established names when they are adopted. The length 30 character names for algorithms shall be post-padded with blank characters. That is if the name is actually "Combination_Method_002", PF.Channlzn.Hist.Comb.Alg(2,m) would be set to "Combination_Method_002 ".

For rules on synchronizing the *m* channelization across data sets and PF.Channlzn.*, see discussion of channelization history in 10.1.3.1

Governing quality metrics

See Also: PF.Channlzn.Hist.Comb.Code.*m*(j)

20.3.15 PF.Channlzn.Hist.WF_Dat(j) [L2]

Brief Definition: PF.Channlzn.Hist.WF_Dat(j) provides parameters for combining waveform

[150] This not completely redundant with MD.ProcHist.L*n*App.Name data. The App may incorporate more than just the channel combination methodology.

	data to create combined channels at the waveform level within a lidar data set.
indices	Index, j, ranges from 1 to 4
Units/representation	Dimensionless, UInt16
Treatment/usage	PF.Channlzn.Hist.WF_Dat(j) provides *counts* of various items. When PF.Channlzn.Hist.Comb.Code.*m*(j)>2[151], and an enterprise level-2 data set contains waveform data that represents the result of a channel combination operation, PF.Channlzn.Hist.WF_Dat(j) can be used to document the relationship between input waveforms and output waveforms. PF.Channlzn.Hist.WF_Dat(1) = the minimum number of input waveform fast time vectors combined to form any of the output waveform vectors. PF.Channlzn.Hist.WF_Dat(2) = the maximum number of input waveform fast time vectors combined to form any of the output waveform vectors. PF.Channlzn.Hist.WF_Dat(3) = the maximum time interval within which all input waveform vectors for any output channel combined waveform occur. PF.Channlzn.Hist.WF_Dat(4) = the maximum spatial interval containing the input waveform look directions for any output channel combined waveform vector. This is given in 'single pixel waveform detector' solid angle units (the solid angle FOV for a single detector is some number of steradians). That is, PF.Channlzn.Hist.WF_Dat(4) is the number of times the solid angle of view for a single input waveform response vector that gives the upper bound for solid angle of integration to form a combined output channel. This value is not necessarily the same as PF.Channlzn.Hist.WF_Dat(2) for two reasons. First, PF.Channlzn.Hist.WF_Dat(2) represents accumulation over both temporal and spatial intervals while PF.Channlzn.Hist.WF_Dat(4) is spatial accumulation only. Second, the look region coverage from which candidate input waveform vectors may be taken can exceed the solid angle actually covered by the selected input vectors, that is, the input vectors might not fill the allowed region for channel combination.
Governing quality metrics	
See Also	PF.Channlzn.Hist.Comb.Code.*m*(j)

[151] This indicates a channel combination that makes the physical meaning of the content of an output channel different than the inputs.

PF.CommonSigma.*

PF.CommonSigma.* parameters describe the aggregation of point into groups given common error location data

20.3.16 PF.CommonSigma.AggLevel [L2,3,4,5]

Brief Definition: PF.CommonSigma.AggLevel is code indicating the level of point grouping at which pre-computed common point error covariance data is supplied.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: While this CMMD supports error propagation for any point, when it can be anticipated that there is a small set of error covariance matrix values for the large number of points in a data set, it is both feasible and convenient to pre-compute the point error covariance and store this as metadata in the data set. When this has been done, the values of the error covariance matrices will be stored in ES.Point.SigmaXYZ(j,k). These values are associated with points via tables which indicate which range of points shares a particular common value of the point covariance matrix. There is also an exceptions table. When a point is in an exceptions table range, it may have a value different than the one for the broad range of points in which it is embedded. PF.CommonSigma.AggLevel is an informational item about the long blocks of points anticipated to have a common value of sigmaXYZ. PF.CommonSigma.AggLevel characterizes the level of aggregation over which the broad point ranges that have a common point covariance matrix value are logically clustered. When the data set is populated with these common values reflecting anticipations based on structural patterns in the uncertainty support data, the appropriate code from the follwing table will be selected for PF.CommonSigma.AggLevel.

code	case
1	pulse level sigmaXYZ reported
2	scan level sigmaXYZ reported
3	subswath level sigmaXYZ reported
4	tile level sigmaXYZ reported
5	swath level sigmaXYZ reported

Governing quality metrics: This should have the correct value populated, but this is a convenience item to support understanding and lack of it is relatively benign.

See Also: ES.Point.SigmaXYZ(j,k) PD.Point.RangeListC.*OrdIndx*(j,k)

PF.Grid.*

PF.Grid.* parameters describe the grid for regularly sampled lidar data

20.3.17 PF.Grid.DelR [L2,3,4,5]

Brief Definition	PF.Grid.DelR is the range increment for intermediate data given on a range grid to create the current data set, if used.
indices	none
Units/representation	Meters Float
Treatment/usage	In the processing of data from (enterprise) level-1 to level-2, it is a possible choice to create an intermediate angle, angle, range grid of data. Quantization to such range grid spacing could have lingering effects in the data which could only be understood by knowing the size of such an increment, if it had been used. Thus, we define PF.Grid.DelR to document such a choice. If, the processing had not used a range grid, this item would not be appropriate to populate. Level-2 and higher point cloud data will use exclusively local Cartesian frames, so PF.Grid.DelR is not to be considered a parameter of the Wide band data grid for enterprise level-2 or higher data.
Governing quality metrics	
See Also	

PF.Overlap.*

PF.Overlap.* parameters describe the overlaps generated by the processor in creating subswaths. Collected overlaps will be found in SC.Overlap.*

20.3.18 PF.Overlap.SubSwath.Dat [L2]

Brief Definition	PF.Overlap.SubSwath.Dat provides the nominal percentage of subswath area in processing generated overlaps
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	Per the discussion of 10.1.4.4, the processor that converts[152] enterprise level-1 to level-2 may incorporate subswath to subswath registrations performed using the overlap between spatially adjacent scans.

[152] Some legacy applications may only have a transient level-1 internal presence and be stovepiped to level-0 data.

PF.Overlap.SubSwath.Dat characterizes the 'as processed' state for subswath overlaps.
PF.Overlap.Subswath.Dat applies only to processor generated subswath overlap for subswath to subswath registration and is the nominal (averaged over all subswath pairs in the data set and rounded to nearest percent) percentage of area of each subswath in the utilized overlap region. For collected subswath overlap, see SC.Overlap.SubswathPrcnt. Consider the plane that best fits the data in a given subswath volume. For each subswath pair, the percentage overlap is twice the common projected area in each plane divided by the sum of the area of the projected planes within the scan volumes.

$$Overlap = \frac{2 \cdot \mathrm{Pr}ojected_Area_{common}}{\mathrm{Pr}ojected_Area_1 + \mathrm{Pr}ojected_Area_2}$$

The projection referred to here is to a common (likely mean) plane for the two subswaths.
So, two adjacent subswaths with projected data fit plane area of 100 sq units each, each having 2 square units in common with its overlap mate, have 2 percent overlap and we would set PF.Overlap.SubSwath.Dat= 2 if all pairs in the data set were like this pair.

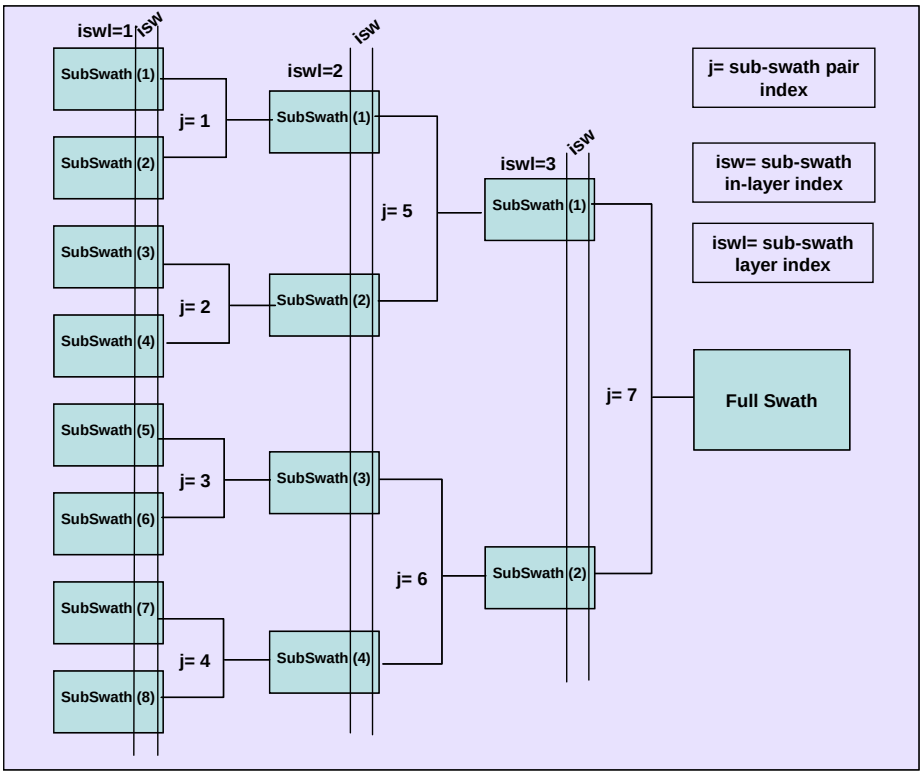
Governing quality metrics

See Also

SC.Overlap.SubswathPrcnt

20.3.19 PF.Overlap.SubSwath.ID1(j) [L2]

Brief Definition	PF.Overlap.SubSwath.ID1(j) identifies the "first" subswath in each subswath pair involved in subswath to subswath registration.
indices	Index, j ranges from PD.LoIndx.Overlap(4) to PD.HiIndx.Overlap(4), that is, nominally from 1 to the number of subswath pairs (over all subswath levels) registered in building the current data set.
Units/representation	Dimensionless, UInt32
Treatment/usage	When mosaicking subswaths together to produce a larger subswath or a full swath (see 10.1.3.3), the adaptive process of registering adjacent subswaths, if performed, is a binary process. That means a "first" and a "second" subswath are registered, meaning the translation and rotation necessary to bring one into relative correspondence with the other is determined. The scheme for deciding which subswath in a pair is first or second is of no consequence. We need only report the (possibly arbitrary) decision. So the subswaths can be considered to be elements of a pair of ordered lists with indices, "isw" (subswath index) and "iswl" (subswath layer index) ranging from the first to last subswath in a layer and from the first to last subswath layer respectively. The identifier for each subswath is its value of "isw" and "iswl" on those lists. The identifiers for the subswaths constituting the jth pair are [PF.Overlap.SubSwath.ID1(j) and PF.Overlap.SubSwath.IDL1(j)] and [PF.Overlap.SubSwath.ID2(j) and PF.Overlap.SubSwath.IDL2(j), with [PF.Overlap.SubSwath.ID1(j) and PF.Overlap.SubSwath.IDL1(j)] being the identifier components for the "first" subswath of the pair. So, the current tag, PF.Overlap.SubSwath.ID1(j), is the value of "isw", subswath index to be associated with a subswath layer index, "iswl" (found in PF.Overlap.SubSwath.IDL1(j)), to uniquely identify the first processing subswath at a particular layer in the jth subswath pair registered in building up the final swath. The figure below illustrates the subswath pair component labeling scheme. In the illustration, the j index over swath pairs ranges from 1 to 7. The values of PF.Overlap.SubSwath.ID1(j) are [1,3,5,7,1,3,1]. The specific subswath is not known until the values of PF.Overlap.SubSwath.IDL1(j), which in this example would be [1,1,1,1,2,2,3] are paired with PF.Overlap.SubSwath.ID1(j).



Tournament bracket diagram of subswath levels

The content of the subswaths indexed by "isw" at each layer index, "iswl" is provided in terms of scans by PF.SubSwath.ScansList.*iswl*(isw,j).

Governing quality metrics: Data must be correct value.

See Also: PD.HiIndx.Overlap(j) PF.Overlap.SubSwath.IDL1(j) PF.SubSwath.ScansList.*iswl*(isw,j)

20.3.20 PF.Overlap.SubSwath.ID2(j) [L2]

Brief Definition: PF.Overlap.SubSwath.ID2(j) identifies the "second" subswath in each subswath pair involved in subswath to subswath registration.

indices: Index, j ranges from PD.LoIndx.Overlap(4) to PD.HiIndx.Overlap(4), that is, nominally from 1 to the number of subswath pairs registered in building the current data set.

Units/representation: Dimensionless, UInt32

Treatment/usage: When mosaicking subswaths together to produce a larger subswath or a full swath (see 10.1.3.3), the adaptive process of registering adjacent subswaths, if performed, is a binary process. That means a "first" and a "second" subswath are registered, meaning the translation and rotation necessary to

bring one into relative correspondence with the other is determined. The scheme for deciding which subswath in a pair is first or second is of no consequence. We need only report the (possibly arbitrary) decision. So the subswaths can be considered to be elements of a pair of ordered lists with indices, "isw" (subswath index) and "iswl" (subswath layer index) ranging from the first to last subswath in a layer and from the first to last subswath layer respectively. The identifier for each subswath is its value of "isw" and "iswl" on those lists. The identifiers for the subswaths constituting the jth pair are [PF.Overlap.SubSwath.ID1(j) and PF.Overlap.SubSwath.IDL1(j)] and [PF.Overlap.SubSwath.ID2(j) and PF.Overlap.SubSwath.IDL2(j), with [PF.Overlap.SubSwath.ID2(j) and PF.Overlap.SubSwath.IDL2(j)] being the identifier components for the "second" subswath of the pair. So, the current tag, PF.Overlap.SubSwath.ID2(j), is the value of "isw", subswath index to be associated with a subswath layer index, "iswl" (found in PF.Overlap.SubSwath.IDL2(j)), to uniquely identify the second processing subswath at a particular layer in the jth subswath pair registered in building up the final swath. The figure in the detail paragraph for PF.Overlap.SubSwath.ID1(j) illustrates how the pair counting index j runs over all subswath layers while subswath index values may be reused over diminishing ranges across increasing layer index values for subswaths.

Governing quality metrics: Data must be correct value.

See Also: PD.HiIndx.Overlap(j) PF.Overlap.SubSwath.IDL1(j) PF.SubSwath.ScansList.*iswl*(isw,j) PF.Overlap.SubSwath.ID1(j)

20.3.21 PF.Overlap.SubSwath.IDL1(j) [L2]

Brief Definition: PF.Overlap.SubSwath.IDL1(j) gives the subswath **layer** value that defines first subswaths in subswath to subswath registration operations.

indices: Index, j ranges from PD.LoIndx.Overlap(4) to PD.HiIndx.Overlap(4), that is, nominally from 1 to the number of subswath pairs registered in building the current data set.

Units/representation: Dimensionless, UInt8

Treatment/usage: When mosaicking subswaths together to produce a larger subswath or a full swath (see 10.1.3.3), the adaptive process of registering adjacent subswaths, if performed, is a binary process. That means a "first" and a "second" subswath are registered, meaning the translation and rotation necessary to bring one into relative correspondence with the other is determined. The scheme for deciding which subswath in a pair is first or second is of no consequence. We need only report the (possibly arbitrary) decision. So the subswaths can be considered to be elements of a pair of ordered lists with indices, "isw" (subswath index) and "iswl" (subswath layer index) ranging

from the first to last subswath in a layer and from the first to last subswath layer respectively. The identifier for each subswath is its value of "isw" and "iswl" on those lists. The identifiers for the subswaths constituting the jth pair are [PF.Overlap.SubSwath.ID1(j) and PF.Overlap.SubSwath.IDL1(j)] and [PF.Overlap.SubSwath.ID2(j) and PF.Overlap.SubSwath.IDL2(j), with [PF.Overlap.SubSwath.ID1(j) and PF.Overlap.SubSwath.IDL1(j)] being the identifier components for the "first" subswath of the pair. So, the current tag, PF.Overlap.SubSwath.IDL1(j), is the value of "iswl", subswath layer index to be associated with a subswath index, "isw" (found in PF.Overlap.SubSwath.ID1(j)), to uniquely identify the first processing subswath at a particular layer in the jth subswath pair registered in building up the final swath. The figure in the detail paragraph for PF.Overlap.SubSwath.ID1(j) illustrates how the pair counting index j runs over all subswath layers while subswath index values may be reused over diminishing ranges across increasing layer index values for subswaths.

Governing quality metrics

See Also PF.Overlap.SubSwath.ID1(j) PD.HiIndx.Overlap(4) PF.SubSwath.ScansList.*iswl*(isw,j)

20.3.22 PF.Overlap.Subswath.IDL2(j) [L2]

Brief Definition PF.Overlap.SubSwath.IDL2(j) gives the subswath **layer** value that defines second subswaths in subswath to subswath registration operations.

indices Index, j ranges from PD.LoIndx.Overlap(4) to PD.HiIndx.Overlap(4), that is, nominally from 1 to the number of subswath pairs registered in building the current data set.

Units/representation Dimensionless, UInt8

Treatment/usage When mosaicking subswaths together to produce a larger subswath or a full swath (see 10.1.3.3), the adaptive process of registering adjacent subswaths, if performed, is a binary process. That means a "first" and a "second" subswath are registered, meaning the translation and rotation necessary to bring one into relative correspondence with the other is determined. The scheme for deciding which subswath in a pair is first or second is of no consequence. We need only report the (possibly arbitrary) decision. So the subswaths can be considered to be elements of a pair of ordered lists with indices, "isw" (subswath index) and "iswl" (subswath layer index) ranging from the first to last subswath in a layer and from the first to last subswath layer respectively. The identifier for each subswath is its value of "isw" and "iswl" on those lists. The identifiers for the subswaths constituting the jth pair are [PF.Overlap.SubSwath.ID1(j) and PF.Overlap.SubSwath.IDL1(j)] and [PF.Overlap.SubSwath.ID2(j) and PF.Overlap.SubSwath.IDL2(j), with

[PF.Overlap.SubSwath.ID2(j) and PF.Overlap.SubSwath.IDL2(j)] being the identifier components for the "second" subswath of the pair. So, the current tag, PF.Overlap.SubSwath.IDL2(j), is the value of "iswl", subswath layer index to be associated with a subswath index, "isw" (found in PF.Overlap.SubSwath.ID2(j)), to uniquely identify the second processing subswath at a particular layer in the jth subswath pair registered in building up the final swath.[153] The figure in the detail paragraph for PF.Overlap.SubSwath.ID1(j) illustrates how the pair counting index j runs over all subswath layers while subswath index values may be reused over diminishing ranges across increasing layer index values for subswaths.

Governing quality metrics

See Also PF.Overlap.SubSwath.ID1(j) PD.HiIndx.Overlap(4) PF.SubSwath.ScansList.*iswl*(isw,j)

PF.Point.*

PF.Point.* parameters describe the properties of points created by processing, especially for processing multiple returns into synthetic points. Coordinated with e.g., PD.TileMap.PtType(j_tile)

20.3.23 PF.Point.IrayNum.*TileNum*(ipt, j) [L2]

Brief Definition PF.Point.IrayNum.*TileNum*(ipt, j) identifies the focal plane detector array and element that a lidar point was collected with.

indices The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

The vertical tile, PF.Point.IrayNum.*TileNum*(ipt, j), can be thought of as a column vector with rows indexed by point index, "ipt". The number of rows in the tile specified by *TileNum* is PD.TileMap.N_rows(j_tile). Each row of such a vertical tile corresponds to a single point in the point cloud carried by the full data set. Like all point oriented tiles defined in this CMMD, there is a row element in some tile for each separate point in the data set. In a sense, ipt is a global index that ranges from 1 to the number of points in a data set and would count from 1 to the total number of points as we cross from tile to tile of point data. Also, the row identifier, "ipt" is just an index and not a

[153] Presumably, both subswaths in a pair come from the same layer, so this item is technically redundant with PF.Overlap.SubSwath.IDL1(j). This is defined for clarity, but enforcement of population is not necessary if the s/w being certified is written to use PF.Overlap.SubSwath.IDL1(j) in place of the current item.

populated data item. Any single tile of data only has local values of ipt (notated by T_ipt elsewhere in this document) that are assigned arbitrarily by the software that stores or reads the tile arrays. For more discussion of the ipt index, see 10.2.1. The T_iptth element of **TileNum*.* for a single value of the string, *TileNum*, provides a row element (i.e. a column entry) for a row in a database. All the **TileNum*(ipt,*) contribute separate elements to that row. Consistency and uniqueness of data order are required to keep point data aligned across all point oriented arrays sharing the same *TileNum*. That is, all populated point arrays, such as ES.Point.PulseNum.*TileNum*(ipt, j) and LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) that share the same value of *TileNum* must have the same points carried in the same order. This is illustrated in figure 10.2.1-1.

The index j has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.ArrayNum.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.ArrayNum.*TileNum*.Flg >0, i.e. when repetition factors are used, there are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. The association between the j_tileth tile and the embedded string that gives the *TileNum* to that tile is supplied by PD.TileMap.TileNum(j_tile).

Units/representation
Dimensionless, Variable Int (type provided by PD.Wordsize.Iray)

Treatment/usage
See discussion of defining rays in 10.1.4. PF.Point.IrayNum.* is a set of tiles that carry the information of which focal plane element and array number (unique to a choice of spectral band and/or polarizer acted as the detector for the point described by global “row” ipt in the collection of point oriented tiles. The value of PF.Point.IrayNum.*TileNum*(ipt, j) for a specific point is drawn from one of the values of the ray index, iray. The array and element associated with a specific iray value are given in SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray), which must be populated if PF.Point.IrayNum.*TileNum*(ipt, j) is populated.

ES.Point.ChNum.*TileNum*(ipt) is the more generally used field. PF.Point.IrayNum.*TileNum*(ipt, j) is a specifically L2 field used for diagnostic work that traces a point not only to a channel, but to a specific array element. **Most product profiles would not likely populate**

PF.Point.IrayNum.*TileNum*(ipt, j).

Governing quality metrics	Value must be correct assignment of array to point.
See Also	PD.TileMap.N_rows(j_tile) PD.TileMap.TileNum(j_tile) SC.Array.FilterCode(ia) PD.IntTileMap.ArrayNum.*TileNum*.Flg SC.WF.RayDesc.ArryGlbl(iray) SC.WF.RayDesc.ElmtGlbl(iray) PD.Wordsize.Iray

20.3.24 PF.Point.SenFrmRefPulse(j_tile) [L2,3,4,5]

Brief Definition	PF.Point.SenFrmRefPulse(j_tile) provides the IPP number that indicates the **pulse** index value for the **ref**erence sensor position used for the j_tileth tile when computing **sen**sor **fr**a**m**e point positions.
indices	The index, j_tile, corresponds to unique instantiations of *TileNum* as provided by PD.TileMap.*TileNum*(j_tile) and ranges from PD.LoIndx.Point.Tile to PD.HiIndx.Point.Tile, that is, over the point tiles.
Units/representation	Dimensionless, UInt32
Treatment/usage	For enterprise level-2 and above, the lidar wideband data will typically include, if not consist solely of, real or synthetic lidar points. This CMMD permits that the points in a data set be in either an earth frame in Cartesian coordinates, or in the sensor frame as described in ES.Swath.SensorBasedCoordFlg(j). When the points of a tile are in the sensor frame, the origin of the coordinate frame is the sensor position. To allow the possibility that a tile bearing sensor frame points can include points from multiple pulses, we supply the pulse number that was the basis of computing the coordinate values given in LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) or LD.Point.Pos.*TileNum*(ipt,j). Consider the process of building a point cloud in the sensor frame over an interval of several pulses. We wish to put this short interval point cloud into a single tile. If the tile accumulates points over an interval long compared to the time the sensor moves a range spacing interval, points at different earth frame positions can pile up at fixed positions in the sensor frame. This means that a tile's data could appear distorted in a sensor frame display. There are two options to assure that a sensor frame point cloud in a single tile can be readily interpreted as a 3-D scene rendering. • Option1) compute the points in the sensor frame on a pulse basis. Collect the points over the pulses in a tile, but adjust the positions based on the sensor position for each pulse by adding the sensor 'pulse

to pulse' offset to each point. Place the tile coordinate frame origin at the sensor position at a reference pulse such as the start/center of the tile integration interval. Then there is a single origin for each tile, always the sensor position, which is always (0,0,0). The scene is undistorted in each tile. The sequence of tiles is a roving eye video. Conversion to earth frame can only be done on a tile sensor reference position basis, as the stored xyz's are not based on the sensor position for any pulse other than the reference pulse. [Effectively this reference sensor position point cloud is in the earth frame instantaneously centered at the sensor position on the reference pulse for the tile. This is how the motion effect is removed from the multi-pulse tile.] Single pulse tiles would make the tile sensor position be the same as the sensor pulse position. The presence of PF.Point.SenFrmRefPulse(j_tile) would indicate that this is the case, as well as supply the data needed to convert the point cloud to the earth frame.

- Option 2) If the PRF is high compared to the ratio (sensor speed/point spacing), coordinate values can be found in the sensor frame and accumulated without compensation for sensor movement over several pulses to build up the points in a tile. We would have to save the pulse number on each point to support eventual full accuracy conversion to earth frame data. A tile would consist of points collected within a narrow time slice.

The CMMD permits either option. When ES.Swath.SensorBasedCoordFlg(j) is given a non-zero value, populating PF.Point.SenFrmRefPulse(j_tile) indicates option 1 is in effect. Not providing PF.Point.SenFrmRefPulse(j_tile) indicates option 2 is in effect.

Governing quality metrics

See Also PD.TileMap.T_Range(j_tile,k) PD.TileMap.PulseRange(j_tile,k) ES.Swath.SensorBasedCoordFlg(j) LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) LD.Point.Pos.*TileNum*(ipt,j)

20.3.25 PF.Point.SynBspread.*TileNum*(ipt) [L2,3,4,5]

Brief Definition PF.Point.SynBspread.*TileNum*(ipt) indicates the point level degree of spectral spread represented by a synthetic lidar point.

indices The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

TileNum, though part of a tag name, acts somewhat as an index. It indicates

which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a "tile". This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Meters^{-1}, float

Treatment/usage: The creation of a synthetic lidar point could, in principle, include the composition of a synthetic point from lidar points determined in different spectral bands. The presence of lidar points in different spectral bands might be associated with, for example, a lidar designed to respond to different wavelengths such as when differential absorption is a key lidar measurement. For such synthetic points, the spectral band spread of a specific point based on points from different bands is given at the point level by PF.Point.SynBspread.*TileNum*(ipt). This is the value of (k_{max}-k_{min}) where k_{min} is the wave number ($2\pi/\lambda$) for the longest wavelength band contributing to the synthetic point and k_{max} is the wave number for the shortest wavelength band contributing to the synthetic point.

Governing quality metrics

See Also

20.3.26 PF.Point.SynPspread.*TileNum* (ipt) [L2,3,4,5]

Brief Definition: PF.Point.SynPspread.*TileNum*(ipt) indicates the point level degree of polarization spread represented by a synthetic lidar point.

indices: The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3 . The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: 1.4 Degree quantization steps, UInt8[154]

Treatment/usage: The creation of a synthetic lidar point could, in principle, include the composition of a synthetic point from lidar points determined in different polarization orientations. The presence of lidar energy received in different

[154] Rounding 360/255 results in polarization spreads measured in 1.4 degree steps. This point level item does not merit doubling the bit volume to get to finer granularity. Polarization filters that are 1.4 degrees wide are expensive and all this is supposed to do is indicate how close in polarization all the samples in a synthesized lidar point derived from a polarization diverse lidar point cloud are. This should be a weighted std deviation and could vary widely from point to point.

polarization orientations might be associated with, for example, a lidar designed to respond to different polarizations, such as when polarization modulation of the transmitted pulse is used to implement pulse compression. For such synthetic points, the polarization spread of a specific point based on points from different polarization returns is given at the point level by PF.Point.SynPspread.*TileNum*(ipt). This is the value of (θ_{max}-θ_{min}) where θ_{max} is the electric vector orientation for the most out of plane-of-incidence polarization contributing to the synthetic point and θ_{min} is the electric vector orientation for the least out of plane-of-incidence polarization contributing to the synthetic point. Note that this value can be computed without defining the orientation of the plane of incidence. Only consistency is needed for this item to be meaningful. An alternative way that multiple polarizations could have entered into the formation of the synthetic points is that the lidar was designed to measure the stokes vector and possessed multiple receive polarization filters, creating separate lidar point clouds for each before points were synthesized.

Governing quality metrics

See Also

20.3.27 PF.Point.SynTasymtry.*TileNum* (ipt) [L2,3,4,5]

Brief Definition

PF.Point.SynTasymtry.*TileNum*(ipt) is the time from the start of the interval supporting a synthetic point to the time of the IPP to which the synthetic point is assigned.

indices

The index, ipt, is a unique row identifier for the virtual database represented by the vertical tiles in a data set. Only consistency and uniqueness are required and ipt does not represent an index in an array which might have local sub-array data population supported such as a sub-matrix of a large matrix. Thus, unlike most other indices used in this document, the range of the index is not provided in this table cell.There is a row element in every point oriented tile in a lidar data set for each separate point. In a sense, ipt is a global index that ranges from 1 to the number of points in a data set and would count from 1 to the total number of points as we cross from tile to tile of point data. Also, the row identifier, “ipt” is not used as a data value. Any single tile of data only has values of ipt (notated by T_ipt elsewhere in this document) that are assigned arbitrarily by the software that stores or reads the tile arrays. The number of rows of data in a tile are given by PD.TileMap.N_rows(j_tile) The association between the j_tileth tile and the embedded string that gives the *TileNum* to that tile is supplied by PD.TileMap.TileNum(j_tile) The T_iptth element of **TileNum*.* for a single value of the string, *TileNum*, provides a row element for a row in a database.

All the *TileNum(ipt,*) contribute separate elements to that row. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3.

Units/representation: Seconds, float

Treatment/usage: To completely describe the relationship of a synthetic point to the time interval that includes all of the measurements that contribute to the synthetic point, we need to include not only the IPP number to which the point is assigned and the duration of the temporal support interval, but also how long into the interval the assigned IPP occurs. Since the assigned pulse time is not required to be at the center of the temporal support interval, we have named this parameter PF.Point.SynTasymtry(ipt) to indicate the general case of asymmetric placement of a synthetic point within the contributing collection interval. It is defined as the time in seconds from the start of the support interval until the time of the IPP number assigned to the synthetic point of interest. Note that this value is permitted to vary for each point.

Note also that when multiple dwells are used in creating a synthetic point, a misleading large difference between the time interval from the earliest contributor to the time tag and the time interval from the time tag to the latest contributor may occur. The difference in line of sight directions needs to be directly analyzed as it can not be inferred to be large from time spreads and asymmetries alone.

Further note that it is intended that when PF.Point.SynTasymtry.*TileNum*(ipt) appears in an enterprise level-4 product, it is inherited from single collection based calculations. If a cross collection level of point synthesis occurs, new temporal support metadata items will be defined and populated.

Governing quality metrics

See Also: PF.Swath.SynTasymtry

20.3.28 PF.Point.SynTspread.*TileNum* (ipt) [L2,3,4,5]

Brief Definition: PF.Point.SynTspread.*TileNum*(ipt) gives the point level duration of the interval contributing to the creation of a synthetic lidar point.

indices: The index, “ipt”, is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3. . The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only

	the column was given as an array index.
Units/representation	Seconds, float
Treatment/usage	PF.Point.SynTspread.*TileNum*(ipt) is the magnitude of the time difference between the first contributing IPP to a synthetic lidar point and the last IPP that contributes to the synthetic point. The significance of PF.Point.SynTspread.*TileNum*(ipt) can be assessed by comparing it to the decorrelation times implied by the correlation decay model with parameters provided by ES.Scan.CorrParms*. Proper processing should not permit point synthesis to include contributions from highly decorrelated measurements if the decorrelation leads to uncertainties equal to or larger than the intended synthetic point voxel spacing or the spatial extent of contributing lidar points. Taken together with PF.Point.SynTasymtry.*TileNum*(ipt), PF.Point.SynTspread.*TileNum*(ipt) gives the key characteristics of synthetic points created by a process that groups spatially neighboring points collected in multiple pulses of a lidar into single output points.
Governing quality metrics	
See Also	PF.Point.SynTasymtry.*TileNum*(ipt)

20.3.29 PF.Point.SynXspread.*TileNum* (ipt) [L2,3,4,5]

Brief Definition	PF.Point.SynXspread.*TileNum*(ipt) gives the point level x-direction spread in lidar points contributing to a single synthetic lidar point.
indices	The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile). *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.
Units/representation	Meters, float
Treatment/usage	If a tile consists of synthetic lidar points (as can be determined by checking the value of PD.TileMap.GlblPtTyping, and if necessary, PD.TileMap.PtType.*TileNum*), a number of neighboring measured lidar points may have been consolidated into an output point. PF.Point.SynXspread.*TileNum*(ipt) reveals the x-direction extent of that neighborhood for a specific point. The orientation of the x-direction cited here is that of the Cartesian frame for the tile containing the point.
Governing quality metrics	

See Also PD.TileMap.GlblPtTyping PD.TileMap.PtType.*TileNum*

20.3.30 PF.Point.SynYspread.*TileNum* (ipt) [L2,3,4,5]

Brief Definition PF.Point.SynYspread.*TileNum*(ipt) gives the point level y-direction spread in lidar points contributing to a single synthetic lidar point.

indices The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3. . The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation Meters, float

Treatment/usage If a tile consists of synthetic lidar points (as can be determined by checking the value of PD.TileMap.GlblPtTyping, and if necessary, PD.TileMap.PtType.*TileNum*), a number of neighboring measured lidar points may have been consolidated into an output point.
PF.Point.SynYspread.*TileNum*(ipt) reveals the y-direction extent of that neighborhood for a specific point.
The orientation of the y-direction cited here is that of the Cartesian frame for the tile containing the point.

Governing quality metrics

See Also PD.TileMap.GlblPtTyping PD.TileMap.PtType.*TileNum*

20.3.31 PF.Point.SynZspread.*TileNum* (ipt) [L2,3,4,5]

Brief Definition PF.Point.SynZspread.*TileNum*(ipt) gives the point level z-direction spread in lidar points contributing to a single synthetic lidar point.

indices The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figure 10.2.1-3 . The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation Meters, float

Treatment/usage If a tile consists of synthetic lidar points (as can be determined by checking

the value of PD.TileMap.GlblPtTyping, and if necessary, PD.TileMap.PtType.*TileNum*), a number of neighboring measured lidar points may have been consolidated into an output point. PF.Point.SynZspread.*TileNum*(ipt) reveals the z-direction extent of that neighborhood for a specific point.
The orientation of the z-direction cited here is that of the Cartesian frame for the tile containing the point.

Governing quality metrics

See Also PD.TileMap.GlblPtTyping PD.TileMap.PtType.*TileNum*

PF.Recon.*

PF.Recon.* parameters describe items related to reconstructing a wideband data set from only the metadata and an archived wideband precursor.

20.3.32 PF.Recon.PrevProdNames(mm,k) [L2,3,4,5]

Brief Definition PF.Recon.PrevProdNames(mm,k) provides the data set names for all precursor data sets in the processing chain for the kth parent data set of the current data set.

indices Index, mm, ranges from PD.LoIndx.PrevStages(k) to Max{PD.HiIndx.PrevStages(k)}, that is, over the previously output data sets for the parent that had the most precursor output data sets.

Index k ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. This provides commonality across the enterprise product levels, but the value of PD.LoIndx.Parents and PD.HiIndx.Parents will always be the same at enterprise level 1, 2, and 3, indicating a single collection operation as the data source for such data sets. Thus, the principal role for index, k, is to support enterprise level-4 fusion products. This strategy permits using the identical tagname for this data at each enterprise level where it may be populated. However, for enterprise level-2 data sets, only index “mm” will have a range of more than one value.

Units/representation Dimensionless, BCS Character String, Length 72

Treatment/usage PF.Recon.PrevProdNames(mm,k) can be populated for enterprise level-2, level-3 and *first generation* level-4 product data sets. A first generation level-4 product is a level-4 with no level-4 parent data sets.

PF.Recon.PrevProdNames(mm,k) provides the data set names (from MD.Meta.DS_Name) inherited from each pre-cursor data set in the

processing chain leading to the current data set. To support looking back more than one precursor generation at level-3 or 4, PF.Recon.PrevProdNames(mm,k) should be populated whenever a data set is generated and it can be used as the basis for inheriting the precursor to be augmented in each new product generation in the processing chain.

Blanks will be used for entries where a particular parent's heritage has fewer precursor products than the parent with the most precursor products.

In the table below, we illustrate the case where :

- PD.LoIndx.PrevStages(k)=1 for all k and Max{PD.HiIndx.PrevStages(k)}=4,
- PD.LoIndx.Parents =1, PD.HiIndx.Parents=3
- Parent data set(1) is the last of a 3 product chain. It's name is *MD.Meta.DS_Name*(1,1)
- Parent data set(2) is the last of a 4 product chain. It's name is *MD.Meta.DS_Name*(1,2)
- Parent data set(3) is the last of a 3 product chain. It's name is *MD.Meta.DS_Name*(1,3)
- *MD.Meta.DS_Name*(mm,k) is the character string supplied by MD.Meta.DS_Name in the data set that is the mmth precursor generation (1→ parent, 2→grand parent, 2→ great grand parent, etc.) of the kth parent.

	k= 1	k= 2	k= 3
mm= 1	*MD.Meta.DS_Name*(1,1)	*MD.Meta.DS_Name*(1,2)	*MD.Meta.DS_Name*(1,3)
mm= 2	*MD.Meta.DS_Name*(2,1)	*MD.Meta.DS_Name*(2,2)	*MD.Meta.DS_Name*(2,3)
mm= 3	*MD.Meta.DS_Name*(3,1)	*MD.Meta.DS_Name*(3,2)	*MD.Meta.DS_Name*(3,3)
mm= 4	72 blank characters	*MD.Meta.DS_Name*(4,2)	72 blank characters

Note that not only does PF.Recon.PrevProdNames(mm,k) identify the family tree for a first generation L-4 or lower level data set, but information such as what the processing stage for each precursor was can be derived since, per the definition of MD.Meta.DS_Name, there are some characteristics encoded into

the data set name. The general rule is that for exploitation, one should use direct metadata rather than parsing the data set name. In this case, we are suggesting that the family tree of data set names might be used for tasks such as product assessment which are PED O&M tasks, not exploitation tasks.

Governing quality metrics

See Also PD.HiIndx.PrevStages(k) PF.Channlzn.Descr.*m*(j) MD.Meta.DS_Name

20.3.33 PF.Recon.App(j) [L1,2,3,4,5]

Brief Definition PF.Recon.App(j) is a unique set of site based[155] identifiers indicating what application(s) were used in the processing chain that created the current data set from raw data.

indices j ranges from PD.LoIndx.Recon(1) to PD.HiIndx.Recon(1), nominally from 1 to the number of product generating applications executed in the processing chain leading to the current data set.

Units/representation Dimensionless, BCS Character 20

Treatment/usage PF.Recon.App(j) is present to document a pedigree of processing steps that permits recreation of a (Enterprise level >1) data set from even its earliest precursor based on only metadata from the product data set. Ideally, the PED will be founded on enterprise level-1 data. So that use of this CMMD need not be delayed until after enterprise adoption of level-1 data sets as roots of the PED, we permit that the initial data set in the processing chain that might be reconstructed is directly documented (see PF.Recon.RootDS(j)) and the chain can then be reconstructed based on that as input and PF.Recon.*. Closely related information is provided in MD.ProcHist.*. That data is principally for discovery of the pedigree of a data set while PF.Recon.* is principally for re-executing the product generation sequence from metadata when the wide band data for more highly processed product levels has been removed from archives to save space.

Two elements, in addition to identifying the algorithm application executed in a step of the processing chain, are necessary to assure uniqueness and the capability to **recon**struct the current data set from its earliest precursor. Those are that the version of the algorithm must be encoded and the site where the application identifier is unique must be known.

To do this, we partition the algorithm name into a 17 character substring and a following 3 character substring. The leading 17 character substring must have a version identifier encoded into it. If the site has not provided an application identifier in this form, the writer for the current data set must

[155] This means that this metadata item reflects information generated at the processing site.

create a conforming string by using key data that might appear in an instantiation of, say, MD.ProcHist.L2App.Versn. The trailing part of the 17 characters may be set to blanks if the application and version are unambiguously coded into less that 17 characters. The last 3 characters are an underscore followed by a 2 character numeric in the range from 00 to 99 that corresponds to the value of j, the index of PF.Recon.Site(j), corresponding to the site where the application indicated in the first 17 characters was executed in the processing chain leading to the current data set.
An example of PF.Recon.App(j) for say j=2 might be "LABxxxCoincdnc052_01" indicating version 52 of the concidence algorithm from laboratory "xxx" was run at the site who value is given by PF.Recon.Site(1).
PF.Recon.App(j) documents the history of processing leading to the current file. Consequently, as products proceed down the processing chain, the range of j increases. Consistency requires that the list of values for PF.Recon.App(j) in derivative products must be populated with the identical sequence of values of those precursor product j value ranges as they were in every precursor product in the processing sequence. In other words, the precursor products have the same PF.Recon.App(j) values in the populated j range as a derivative product. But for precursors, the range of j is less.
Each value of j corresponds to a separate execution of an application. If the same application is run more than once, it will appear in the list with the j values matching when in the sequence it was executed.

For products at enterprise level 3 and 2, the processing order from first to last is to be the convention. Each value of j corresponds to a separate execution of an application. If the same application is run more than once, it will appear in the list with the j values matching when in the sequence it was executed. The order must be matched by that used in PF.Recon.AppDescr(j).

Governing quality metrics

See Also PD.HiIndx.Recon(j) PF.Recon.Site(j) MD.ProcHist.L2App.Versn PF.Recon.RootDS(j)

20.3.34 PF.Recon.AppDescr(j) [L1,2,3,4,5]

Brief Definition PF.Recon.AppDescr(j) bears text descriptions of the applications in the processing chain leading to the current data set.

indices j ranges from PD.LoIndx.Recon(1) to PD.HiIndx.Recon(1), nominally from 1 to the number of product generating applications executed in the processing chain leading to the current data set.

Units/representation Dimensionless, BCS Character (100)

Treatment/usage	PF.Recon.AppDescr(j) provides a brief characterization of the key processing function performed by the individual applications in the processing chain leading to the current product. When populating PF.Recon.AppDescr(j) at enterprise level-2 or level-3, the order of j indexed entries will be the processing chain order, matching the order used for PF.Recon.App(j) . Each value of j corresponds to a separate execution of an application. If the same application is run more than once, it will appear in the list with the j values matching when in the sequence it was executed.
Governing quality metrics	
See Also	PF.Recon.CommandFiles(j)

20.3.35 PF.Recon.CommandFiles(j) [L1,2,3,4,5]

Brief Definition	PF.Recon.CommandFiles(j) provides the values of input parameters for each of the application executions that led to creation of the current data set.
indices	j ranges from PD.LoIndx.Recon(1) to PD.HiIndx.Recon(1), nominally from 1 to the number of product generating applications executed in the processing chain leading to the current data set.
Units/representation	Dimensionless, BCS Character, variable length provided by PD.WordSize.Recon.
Treatment/usage	For each value of j, PF.Recon.CommandFiles(j) provides a text string constituting a comma delimited set of strings of the form, "*name=value*". The *name* substrings will be instantiations of names of input parameters for the application. The *value* substrings will be the values that were used for the corresponding parameter when the jth application was run in the sequence that led to creating the current data set. If the original application used a name-value input file of this form, PF.Recon.CommandFiles(j) might be an actual image of the inputs file. More likely, a suitable input file to re-execute the same run would have to be regenerated based on the content of PF.Recon.CommandFiles(j) for a particular value of j. For these command files to be consistent with the expectations of this CMMD, it is essential that each execution of a product generating application not only contain the CMMD compliant input data set names, but also the name of the output CMMD compliant product. [refer to MD.Meta.DS_Name].
Governing quality metrics	
See Also	PD.WordSize.Recon MD.Meta.DS_Name

20.3.36 PF.Recon.RootDS(j) [L2,3,4,5]

Brief Definition	PF.Recon.RootDS(j) provides the name and an archive location for the root wide band data set(s) processed to create a level-2 through 4 product.
indices	Index, j, ranges from PD.LoIndx.Recon(3) to PD.HiIndx.Recon(3), covering a range equal to twice the number of root data sets involved in creating the current product. For enterprise level-2, index j takes two values (0:1 or 1:2)
Units/representation	Dimensionless, character 71
Treatment/usage	~~Consider a dummy index, n that ranges over the root data sets for the current product. If the current product is enterprise level-3 or below, only one root data set exists. If the current product is level-4, two or more root data sets exist~~. The index j of PF.Recon.RootDS(j) takes on the values of (2n-1) and 2n in sequence. PF.Recon.RootDS(1) is the Root Data Set name for the processing chain leading to the current data set. (U) For Enterprise product level-2, the root data set names will belong to only level-1 or level-2 data sets. So lidar product type levels listed here are: **LID**ar **L**evel **1**, **LID**ar **L**evel **2.** These are part of the data set name for the root datset. (U) An enterprise level-1 data set name is given by concatenating the following substrings: time(Julian date + time after midnight)-- CCYYDDDHHMMSS.S (15 characters) of file creation platform identifier--(6 characters) Collection Num -- (4 characters) Sensor Identifier --(6 characters) Channel identifier --(2 characters) Prod type "LIDL1" Level-n stage/Form –(2 characters) Generation Site ID –(6 characters) Spacer string "____" (4 characters) GenSite Unique product serial number—(up to 10 characters) DAL synch string "*.CCCCCCCCCC*" e.g ".DALHDF5v02" (11 characters) An enterprise level-2 data set name is given by concatenating the following substrings: time(Julian date + time after midnight)-- CCYYDDDHHMMSS.S (15 characters) of file creation platform identifier--(6 characters) Collection Num -- (4 characters) Sensor Identifier --(6 characters)

Channel identifier --(2 characters)
Prod type "LIDL2"
Level-n stage/Form –(2 characters)
Generation Site ID –(6 characters)
Spacer string "___" (4 characters)
GenSite Unique product serial number—(up to 10 characters)
DAL synch string ".*CCCCCCCCCC*" e.g ".DALHDF5v02" (11 characters)

In the above described substrings:
Platform identifier--(6 characters) refers to a registered platform identifier that uniquely determines the kind and identity of the lidar bus vehicle. The identifier might be composed of 3 char alphanumeric symbol such as "DC3" or "707" and a tail identifier composed of 3 alpha-numerics such as "012" or "QRP" or "N32". This also permits designating using US N-numbers up N999ZZ as platform identifiers.

Collection Num-- (4 characters) This is a place to composite strings such as a sortie number and a project number. For example, two alpha numeric characters each can be used for a sortie and project to build the collection number. May only be unique for a single day, week, month,…

Sensor Identifier --(6 characters) refers to a registered sensor identifier such as "OPK015", this is a unique tag for the individual lidar instrument.

Channel identifier --(2 characters) This is most readily treated as a number such as "01" or "02". If a sensing instrument has more than a single sensing channel and can generate multiple wide band data streams, they should be identified by a sequential number starting with 1 and counting upward. If all channels are present in a data set, the Channel identifier should be set to "00". If there is only one channel in the sensor, the value should be set to "01". The number of channels in a lidar are not large and the 2 digit limit is quite generous.

Prod type The caps fields listed in this data definition, e.g., "LIDL4" should be used as the value of the substring to be concatenated.

Level-n stage/Form –(2 characters) A registered code for the product stage should be used here. Examples of what will be proposed for codes are:
Level-2 forms
S1- noisy point cloud

2a- Noise filtered point cloud using a'priori methods such as coincidence processing of some kind
2b- Scan to scan registered
S2-Waveform clips only
S3- noisy Point clouds with waveform clips
S4- de-noised points with waveform clips

Generation Site ID- This is a registered 6 character identifier for authorized processing sites within the NSG that are authorized lidar processing sites. Examples of strings could be, "NAT002", "GIAT03", "EUT005", The identifiers are alpha-numeric and unique for each processing site

GenSite Unique product serial number—Each generation site creates a serial sequence that is unique within a period of time that can be encoded as an interval beginning on a value of CCYYDDD. Up to 10 characters can be used and the active part of any string using less than all 10should have the needed number of zeros pre-pended to created the 10 character serial number.

DAL synch string--- The 11 character DAL synch string is recognizable since it always begins ".DAL" The remaining 7 characters [should be registered strings] indicating a unique set of DAL linkable code modules that will successfully ingest data written to the data set. The (in this case, lidar) data set writing application chooses the registered DAL implementation to use when it outputs the data set. It then places the registered string for the intended DAL ingest module set at the end of the parameter whose name is MD.Meta.DS_Name and gives the output file the same name as the value of MD.Meta.DS_Name.

Example MD.Meta.DS_Name value:
2011249113210.1B470031102LKA01201LIDL2S3GIAT03____0000012345.DALHDF5v02

This data set name is a label designed for uniqueness and should not be parsed for file content information. Direct use of LIDL*n* metadata fields is the correct way to obtain that information. Limited human recognizable content is incorporated for facilitation of exploitation work flow, but this data item is not a replacement for actual *Discovery* or *Exploitation* support data.

This metadata item is analogous to NITF's FTITLE, in that it is a name for the data set that is embedded in the data set. Operating systems may append their own suffixes to the data set name, but this acts as a global core

string convention for naming LIDL*n* sets.

PF.Recon.RootDS(2) is the name of an archive location for the Root Data Set named in PF.Recon.RootDS(1). It is recommended that a registry of regulated archive references that can be encoded into 71 or less characters or less be maintained in the operating enterprise so that interpretation of the strings carried by PF.Recon.RootDS(2) can be uniform across the enterprise.

Note that for Enterprise level-2 and 3, and for level-3 based level-5 products, only PF.Recon.RootDS(1) and PF.Recon.RootDS(2) will be populated. They will carry a data set name and the location of where it is archived, respectively, in blank post-padded character strings of length 71.

Governing quality metrics

See Also MD.Meta.DS_Name

20.3.37 PF.Recon.Site(j) [L1,2,3,4,5]

Brief Definition PF.Recon.Site(j) provides the names of sites where applications were executed in the processing chain leading to the current data set.

indices Index, j, ranges from PD.LoIndx.Recon(2) to PD.HiIndx.Recon(2), that is over the all of the facilities where processing was done that lead to the current product.

Units/representation Dimensionless, BCS Character(20)

Treatment/usage PF.Recon.Site(j) is a character string representing the jth cited site where processing was performed in the chain leading to the current data set. When an application from this site is listed in PF.Recon.App(j), the last 2 characters of PF.Recon.App will be the numerical value of the index j for PF.Recon.Site(j) for the site at which the application name was provided. For example, if there are two processing sites, PF.Recon.Site(1)= “Big Place up the rd ” and PF.Recon.Site(2)= “Big Plc dwn the rd ” and three algorithms were used at the two sites, then the embedded site indices in the three PF.Recon.App(j), would point to only to the 2 sites. Suppose that algorithm “Did this 2 data ” was executed second and run at “Big Plc dwn the rd ”, then PF.Recon.App(2) would be set to “Did this 2 data _02”. But we could also have run “Did what 2 data ” at “Big Plc dwn the rd ”. Then the following correspondences would accrue:

	PF.Recon.Site	PF.Recon.App(j)
j=1	PF.Recon.Site(1)= “Big Place up the rd ”	“Did that 2 data _01”.
j=2	PF.Recon.Site(2)= “Big Plc dwn the rd ”	“Did this 2 data _02”.

j=3	PF.Recon.Site(2)= "Big Plc dwn the rd "	"Did what 2 data _02".

Governing quality metrics

See Also PD.HiIndx.Recon(2), PF.Recon.App(j)

PF.Registn.*

PF.Registn.* parameters describe items related to registrations performed by the processing applications in generating a data set

20.3.38 PF.Registn.PulseDat(j) [L2]

Brief Definition PF.Registn.PulseDat(j) provides codes for the status of pulse to pulse registrations, if performed, to composite frames into scans.

indices Index, j =1,2,3

Units/representation Dimensionless, UInt8

Treatment/usage Per 10.1.3, lidars record data at the pulse level. The results of consecutive pulses are typically aggregated into "scans" based on a single iteration of a cyclical control of the beam optical axis. It may be the case that the processor performs pulse to pulse registration in creating these scan data blocks. If the lidar sensor is operated in a mode where the sequence of pulse level data needs to be registered on a pulse to pulse basis, the processor has some options in mosaicking together the pulses to produce scan area sized scene blocks.

For intra-scan pulse to pulse registration, PF.Registn.PulseDat(1) is a utilization code that serves as a complement to adjustment data. Defined code values are:

0--> overlap not used/attempted for registration
1--> overlap used to determine 3-D translation,
2--> overlap used to determine 3-D rotation
3--> overlap used to determine 2-D rotation
4--> overlap used to determin 3-D translation and 3-D rotation
5--> overlap used to determin 3-D translation and 2-D rotation

PF.Registn.PulseDat(2) is a code for how overlap based registration errors computed on basis of pulse to pulse registration errors are treated:

0--> Registration on intra scan data computed ignoring any pulse to pulse registration error (i.e. deterministic data only[156] is used)

1--> Registration on intra scan data computed for pulse to pulse

[156] Either Hypothesis test or a'priori information suggested low confidence, or the signal processing design lacks the capability. PF.Registn.PulseDat(1) may or may not be 0.

registration error corrected data (adaptive data is generated from pulse to pulse overlaps and applied to the point cloud data in compositing the scans)

2--> Registration on intra scan data computed for pulse to pulse registration error application model, but no actual XYZ shifts or rotations were ever applied on point basis.

PF.Registn.PulseDat(3) is a code for the frame in which adjustments in PF.Adjustmts.Mean.Pulse.*TileNum*(ip,j) are expressed.

0→ The adjustments are expressed in the ECEF frame, meaning that the center of the earth is the origin of the vector expressing the rotation axis and that shifts are along the ECEF XYZ axes.

1→ The adjustments are expressed in the local Cartesian frame of the tile containing point data to be corrected. This means that the Cartesian frame origin is the origin of the vector expressing the rotation axis and that shifts are along the Local Cartesian XYZ axes (usually the local ENU system)

Governing quality metrics

See Also PF.Adjustmts.Mean.Pulse(is, j)

20.3.39 PF.Registn.ScanDat(j) [L2,3,4,5]

Brief Definition PF.Registn.ScanDat(j) provides codes for the status of scan to scan registrations, if performed, to composite scans into subswaths.

indices j=1,2,3,4

Units/representation Dimensionless, UInt8

Treatment/usage Per 10.1.3, lidars record data at the pulse level. The results of consecutive pulses are typically aggregated into "scans" based on a single iteration of a cyclical control of the beam optical axis. These in turn may be registered to assemble subswaths. If the lidar sensor is operated in a mode where the sequence of scan level data aggregations needs to be registered, the processor has some options in mosaicking together the scans to produce subswath scene blocks.

For intra-subswath scan to scan registration, PF.Registn.ScanDat(1) is a utilization code that serves as a complement to adjustment data. Defined values are:

0--> overlap not used for registration computation
1--> overlap used to determine 3-D translation,
2--> overlap used to determine 3-D rotation
3--> overlap used to determine 2-D rotation[157]

[157] For purposes of scan to scan registration, 2-D rotation is about an axis that is perpendicular to a target scene base plane. Thus, horizontal rotational misalignments might be detected, but "ground tilt" mismatches can't be detected.

4--> overlap used to determin 3-D translation and 3-D rotation
5--> overlap used to determin 3-D translation and 2-D rotation

PF.Registn.ScanDat(2) is a code for how overlap based registration errors computed on basis of scan to scan registration errors are treated:

0--> Registration to produce subswath data computed ignoring any scan to scan registration error (i.e. deterministic data only is used). This is likely if either a hypothesis test or a'priori information suggested low confidence, or the signal processing design lacks the capability. PF.Registn.PulseDat(1) may or may not be 0.

1--> Registration transform to build subswath data computed from scan to scan registration and applied on subswath basis (adaptive data is generated from collection or processor generated scan to scan overlaps and applied to the point cloud data in compositing the scans to subswaths)

2--> Registration transform to build subswath data computed[158] from scan to scan registration error application model, but no actual XYZ shifts or rotations were ever applied on point basis at the subswath level.

PF.Registn.ScanDat(3) is a code indicating whether the data used to derive geometrical adjustments comes from a single or multiple swaths.

PF.Registn.ScanDat(3) [=0] only *intra*-swath (single collection operation) input to registration,
PF.Registn.ScanDat(3) [=1] only *inter*-swath registration used
PF.Registn.ScanDat(3) [=2] intra-*and* inter-swath registrations used

PF.Registn.ScanDat(4) is a code indicating the hard bound referenced in PF.Adjustmts.Mean.Scan(is, j) for budgeted mean shift residual due to the value of the rotation adjustment "pin" point.

PF.Registn.ScanDat(4) = 0→ 1meter
PF.Registn.ScanDat(4) = 1→ .5 meter
PF.Registn.ScanDat(4) = 2→ .1 meter
PF.Registn.ScanDat(4) = 3→ RMS of trace of covariance for the affected point cloud block
PF.Registn.ScanDat(4) = 4→.5 times RMS of covariance trace for the affected point cloud block
PF.Registn.ScanDat(4) = 5→.1 times RMS of covariance trace for the affected point cloud block

[158] Computed values might be stored for downstream application, or they might be ignored as unreliable. The cases are distinguished by the presence of adjustment data.

Per PF.Adjustmts.Mean.Scan(is, j), the adjustments are expressed in the ECEF frame, meaning that the ECEF frame is the reference for the direction of the rotation axis and that shifts are along the ECEF XYZ axes.

Governing quality metrics

See Also PF.Adjustmts.Mean.Scan(is, j) ES.Origin.GeodtcEpoch(j)

20.3.40 PF.Registn.SubswathDat(iswl,j) [L2,3,4,5]

Brief Definition PF.Registn.SubswathDat(iswl,j) provides codes for the status of scan group to scan group registrations, if performed, to composite subswaths into larger subswaths or the final product swath.

indices Index, iswl, ranges from 1 to PF.Subswath.Nlayers
(or 0: (PF.Subswath.Nlayers-1)), that is over the subswath layers used to generate the current product.
j=1,2,3,4

Units/representation Dimensionless, UInt8

Treatment/usage Per 10.1.3, lidars record data at the pulse level. The results of consecutive pulses are typically aggregated into "scans" based on a single iteration of a cyclical control of the beam optical axis. These in turn may be registered to assemble subswaths. Subswaths of lidar points or synthetic lidar points may, in turn, need to be registered to produce higher level subswaths, and eventually, the final swath. The processor has some options in mosaicking together the subswaths to produce higher level subswath scene blocks or the final swath level product.

The registration of adjacent subswaths may be computed in either of the following ways:

1--> collective scan group to scan group registration of all pairable scans in the overlapping region treated as a single large overlap registration

2--> pair wise registrations performed at the individual scan pair level and results combined. At this stage of building up the swath, the selected coverage at the "scan level" need not be precisely that of any actually collected scan; it may just be approximately that amount of real estate.

For inter-subswath scan to scan registration, PF.Registn.SubswathDat(iswl,1) is a utilization code that serves as a complement to adjustment data. The defined values that apply to either of the above registration approaches are:

0--> overlap not used/attempted for registration

1--> overlap used to determine 3-D translation,

2--> overlap used to determine 3-D rotation
3--> overlap used to determine 2-D rotation
4--> overlap used to determine 3-D translation and 3-D rotation
5--> overlap used to determine 3-D translation and 2-D rotation

PF.Registn.SubswathDat(iswl,2) is a code for how overlap based registration corrections computed on basis of scan to scan registration transform detection are treated:

0--> Registration to mosaick subswaths to larger subswaths computed ignoring any subswath to subswath[see**Error! Reference source not found.** registration rror (i.e. deterministic data only is used)

1--> Registration transform to mosaick subswaths was computed from overlapping subswath sub-block[159] registration(s) and applied to resulting combination of subswaths.

2--> Registration on subswath data computed for overlapping subswath sub-block registration(s), but no actual XYZ shifts or rotations were applied on point basis.

PF.Registn.SubswathDat(iswl,3) is a code indicating whether the data used to derive geometrical adjustments comes from a single or multiple swaths.

PF.Registn.SubswathDat(iswl,3) [=0] only *intra*-swath (single collection operation) input to registration,

PF.Registn.SubswathDat(iswl,3) [=1] only *inter*-swath registration used

PF.Registn.SubswathDat(iswl,3) [=2] intra- *and* inter-swath registrations used

The adjustments are expressed in the ECEF frame, meaning that the center of the earth is the origin of the vector expressing the rotation axis and that shifts are along the ECEF XYZ axes.

PF.Registn.SubswathDat(iswl,4) is a code indicating the hard bound referenced in PF.Adjustmts.Mean.Subswath.*Iswl*(isw, j) for budgeted mean shift residual due to the value of the rotation adjustment "pin" point.

PF.Registn.SubswathDat(iswl,4) = 0→ 1meter
PF.Registn.SubswathDat(iswl,4) = 1→ .5 meter
PF.Registn.SubswathDat(iswl,4) = 2→ .1 meter
PF.Registn.SubswathDat(iswl,4) = 3→ RMS of trace of covariance for

[159] Intended subswath to subswath registration was done on the SubSwathDat(1) basis and applied. This can be based on registration of subswath overlaps ("sub-blocks") that are full or partial scans and can be done on a collective basis, or may average a number of local registrations. For example, adaptive data can be generated from processor generated subswath to subswath overlaps (covering only the parts of scans that are in the subswath overlaps) and applied to the point cloud data in compositing the scans to subswaths.

the affected point cloud block
PF.Registn.SubswathDat(iswl,4) = 4→.5 times RMS of covariance trace for the affected point cloud block
PF.Registn.SubswathDat(iswl,4) = 5→.1 times RMS of covariance trace for the affected point cloud block

Per PF.Adjustmts.Mean.Subswath.*Iswl*(isw,j), the adjustments are expressed in the ECEF frame, meaning that the ECEF frame is the reference for the direction of the rotation axis and that shifts are along the ECEF XYZ axes.

Governing quality metrics

See Also PF.Adjustmts.Mean.Subswath(iswp, j) PF.Recon.AppDescr(j) PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j)

PF.SAL.*

PF.SAL.* parameters describe items related to synthetic aperture processing of the time domain lidar data.

20.3.41 PF.SAL.Autofocus [L2,3,4,5]

Brief Definition PF.SAL.Autofocus indicates the applied autofocus correction degree

indices none

Units/representation Dimensionless, UInt8

Treatment/usage =0 not applied, =1 quadratic only, =2 High Order autofocus
Stub, Details pending, Need to support tree and table data in 9.2

Governing quality metrics

See Also LD.SAL.DataGrid(i,j,k)

20.3.42 PF.SAL.CollMode [L2,3,4,5]

Brief Definition PF.SAL.CollMode indicates the synthetic aperture collection mode

indices Index, j=1 to 3 *ChNum, TileNum* act somewhat as indices

Units/representation Dimensionless, float

Treatment/usage =0 spotlight, =1 strip
Could rename to SC.SAL.CollMode, if we define more SC.SAL parameters. We have already put most of the pulse compression parameters into SC.* to support non-synthetic aperture collection. So these parms which would belong to the synthetic aperture group in a radar case, have a different "class" (2nd dot delimited string).
Stub, Details pending, Need to support tree and table data in 9.2

Governing quality metrics

See Also	LD.SAL.DataGrid(i,j,k)

20.3.43 PF.SAL.DOF [L2,3,4,5]

Brief Definition	PF.SAL.DOF is the depth of focus in meters for the synthetic aperture collection.
indices	none
Units/representation	meters, float
Treatment/usage	*Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.3.44 PF.SAL.ProcGrid.Axes.*ChNum.TileNum*(j) [L2,3,4,5]

Brief Definition	PF.SAL.ProcGrid.Axes.*ChNum*.*TileNum*(j) indicates the processing plane orientation with a unit vector for the range-like direction
indices	Index, j=1 to 3 *ChNum, TileNum* act somewhat as indices
Units/representation	Dimensionless, float
Treatment/usage	j= 1,2,3 ECEF x,y,z range-like axis direction components rest of processing plan orientation given by the normal. *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.3.45 PF.SAL.ProcGrid.Code [L2,3,4,5]

Brief Definition	PF.SAL.ProcGrid.Code indicates whether the processing plane is a slant plane or ground plane.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	=0 for slant plane, =1 for ground plane *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.3.46 PF.SAL.ProcGrid.Normal.*ChNum*.*TileNum* (j) [L2,3,4,5]

Brief Definition	PF.SAL.ProcGrid.Normal *ChNum*.*TileNum*(j) indicates the processing plane orientation with an ECEF unit vector
indices	Index, j=1 to 3 *ChNum,TileNum* act somewhat as indices
Units/representation	Dimensionless, float

Treatment/usage	j= 1,2,3 ECEF x,y,z processing plane normal direction components rest of plane orientation given by the range-like direction unit vector *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

20.3.47 PF.SAL.ProcGrid.GRP.*ChNum*.*TileNum*(j) [L2,3,4,5]

Brief Definition	PF.SAL.ProcGrid.GRP.*ChNum*.*TileNum*(j) indicates the ECEF position of the ground reference profile used in processing the time domain lidar data
indices	Index, j=1 to 7 *ChNum*, *TileNum* acts like an index. The tiles at this point are small enough that the ground reference profile can be represented to sufficient accuracy as a line with a constant spatial step length during the tile specific synthetic aperture.
Units/representation	various, float
Treatment/usage	j= 1,2,3 ECEF x,y,z processing reference position at the first IPP of the time domain data for the current tile (by channel, *ChNum*). This is the position (in meters) against which the range data is stabilized for synthetic aperture processing for the first IPP of a tile. j=4,5,6 ECEF x,y,z unit vector (dimensionless) along a locally linear, tile specific, ground reference profile. If the processing is done with respect to a single point, these entries are zero. j=7 this is the step length between GRP positions at each IPP during the synthetic aperture integration time. Each tile processed for a synthetic aperture covers either a short distance along the GRP or only a point and the output reference point (PF.SAL.ProcGrid.ORP(j) should be a value near that supplied in the first 3 components of PF.SAL.ProcGrid.GRP.*ChNum*.*TileNum*(j). The time increment associated with each step corresponds to the duration of the IPP between successive slow time range vectors. This would be tracked to the specific tile being processed. *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid.*ChNum*.*TileNum*(i,j,k)

20.3.48 PF.SAL.ProcGrid.Type [L2,3,4,5]

Brief Definition	PF.SAL.ProcGrid.Types carries a code to indicate the nature of the processing grid
indices	none
Units/representation	Dimensionless, UInt8

Treatment/usage	PF.SAL.ProcGrid.Type =1 for range, Doppler (PFA), =2 for along, cross track(RMA) processing grid *Stub, Details pending, Need to support tree and table data in 9.2*
Governing quality metrics	
See Also	LD.SAL.DataGrid(i,j,k)

PF.Sort.*

PF.Sort.* parameters describe items related to data sorting operations performed on the data set before it was stored in the current form.

20.3.49 PF.Sort.PointOrder.*TileNum* [L2,3,4,5]

Brief Definition	PF.Sort.PointOrder.*TileNum* is a flag indicating whether points in the tile indicated by *TileNum* are sorted in increasing or decreasing order of the identified sorting parameters
indices	None, Int8
Units/representation	dimensionless
Treatment/usage	PF.Sort.PointOrder.*TileNum* set to a positive value indicates ascending sort. PF.Sort.PointOrder.*TileNum* set to a negative value indicates descending sort.
Governing quality metrics	
See Also	PF.Sort.PointPriority.*TileNum*(j)

20.3.50 PF.Sort.PointPriority.*TileNum*(j) [L2,3,4,5]

Brief Definition	PF.Sort.PointPriority.*TileNum*(j) provides the priorities by which point data are sorted within a given tile.
indices	Index, j, ranges from 1 to 10
Units/representation	Dimensionless, UInt8
Treatment/usage	This CMMD permits the data in each tile to have a unique sorting priority. This is a “sort by, then by” descriptor. The range of j is over the populatable point tile data items. For each value of j corresponding to these point row items j= 1--X 2--Y 3--Z 4-- pulse number 5—channel 6-- intensity 7—return

8—User defined quantity #1
9—User defined quantity #2
10- User defined quantity #3

an integer code is present. If a user defined quantity [see *X*D.UserDefined.PtDat.P*n.TileNum*(ipt)] is added to the data set at the point level and were a sorting basis, PF.Sort.PointPriority.*TileNum*(8, 9,10) could have values other than 0 . When no user defined data is present, PF.Sort.PointPriority.*TileNum*(8, 9,10) will all be set to 0.
Codes are:
0--this item is not sorted
1--This item is first sorting priority
2--This item is first "then by" sorting priority
3--This item is the next "then by" sorting priority
etc thru 7
Only the code, "0" may be repeated. Codes greater than 0- must be consecutive values
Then the data for PF.Sort.PointPriority.*TileNum*(j) might be (3,2,0,4,0,0,1,0,0,0) indicating no sorting on Z, intensity or channel and return being first sorting priority and Y being the second sorting priority.
All point quantities sharing the same *TileNum* must be consistently ordered so that the kth element of all point row members corresponds to the kth point. PF.Sort.PointPriority.*TileNum*(j).
If a product is eventually made where more than 3 user defined items are needed, the CMMD would need to be modified to extend the range of index, j, beyond 10.

Governing quality metrics: When populated, data values must accurately represent the processing and sorting choices made on a tile by tile basis.

See Also: PF.Sort.PointOrder.*TileNum*

20.3.51 PF.Sort.WFClpOrder.*TileNum* [L2]

Brief Definition: PF.Sort.WFClpOrder.*TileNum* is a flag indicating whether waveform clip oriented data in the tile indicated by *TileNum* are sorted in increasing or decreasing order of the identified sorting parameters

indices: None

Units/representation: Dimensionless, Int8

Treatment/usage: PF.Sort.WFClpOrder.*TileNum* set to a positive value indicates ascending sort. PF.Sort.WFClpOrder.*TileNum* set to a negative value indicates descending sort.

Governing quality metrics:

See Also PF.Sort.WFClpPriority.*TileNum*(j)

20.3.52 PF.Sort.WFClpPriority.*TileNum*(j) [L2]

Brief Definition PF.Sort.WFClpPriority.*TileNum*(j) provides the priorities by which waveform clip data are sorted within a given tile.

indices Index, j, ranges from 1 to 7

Units/representation Dimensionless, UInt8

Treatment/usage This CMMD permits the data in each tile to have a unique sorting priority. The waveform clips[LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it)] are sorted by ascending ipi by definition, but ipi is a flexible assignment to the slow time sequential index, ip [see PD.WFClp.PulseNum.*TileNum*(ipi)]. Also, the LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) array is inherently ordered by virtue of index, iray. But "iray" is assigned to arrays via SC.WF.RayDesc.Arry.*TileNum*(iray), which can be rearranged from a level-1 ordering to a level-2 ordering to meeting new sorting objectives. By setting the ordering of focal plane arrays in the definition of iray, the order of the appearance of wavefom clips in a tile may be chosen so as to group by channel attributes such as spatial diversity, polarization or spectral response. PF.Sort.WFClpPriority.*TileNum*(j) is a "sort by, then by" descriptor. The range of j is over the physical characterizations of the waveforms clips in a tile. Waveform clip tiles are return specific at the top level, so return is not an internal tile sorting option. For each value of j corresponding to these pulse-ray row items

j= 1-- time
2—Spatial diversity channel
3—Pol channel
4—Spectral Band channel
5—User defined item #1
6—User defined item #2
7—User defined item #3

an integer code is present. Codes are:
0--this item is not sorted
1--This item is first sorting priority
2--This item is first "then by" sorting priority
3--This item is next "then by" sorting priority
etc thru 7.
Only the code, "0" may be repeated.
Then the data might be (1,0,0,2,0,0,0) indicating no sorting on spatial aspect diversity or polarization and time being first sorting priority with spectral channel being next.
All WFClp quantities sharing the same *TileNum* must be consistently ordered

	so that the each element of all pulse-ray labeled quantities corresponds to the the same waveform clip. If a user defined quantity [see *X*D.UserDefined.PulseDat.P*n.TileNum*(ipt)] is added to the data set at the pulse level and were a sorting basis, PF.Sort.WFClpPriority.*TileNum*(5, 6, 7) could have values other than 0 . When no user defined data is present, PF.Sort.WFClpPriority.*TileNum*(5,6,7) will all be set to 0. If a product is eventually made where sorting on more than 2 user defined items is needed, the CMMD would need to be modified to extend the range of index, j, beyond 7. If PF.Sort.WFClpPriority.*TileNum*(j) is not populated, either the data is irrelevant due to the absence of waveform clips, or using software should treat it as if all elements were set to 0.
Governing quality metrics	When populated, data values must accurately represent the processing and sorting choices made on a tile by tile basis.
See Also	PF.Sort.PointOrder.*TileNum* SC.WF.RayDesc.Arry.*TileNum*(iray) LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) PD.WFClp.PulseNum.*TileNum*(ipi)

PF.Subswath.*

PF.Subswath.* parameters describe items related to subswath processing of the data. These are either in response to subswath collection or "tournament bracket" parallelization of the lidar processing.

20.3.53 PF.Subswath.Nlayers[L2,3,4,5]

Brief Definition	PF.Subswath.Nlayers is the number of subswath layers chosen by the processor to create the current product.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	Per the discussion of 10.1.4.3, an enterprise level-2 lidar processing application has the option of creating subswaths by combining groups of contiguous scans. Further combining such groups in a hierarchical sequence creates intermediate "layers" of subswaths. Each succeeding "layer" has fewer subswaths and the last such layer when combined yields the full product swath. This is most likely to be performed in a parallel processing implementation of lidar point cloud formation. However, if there were collection subswaths due to, for example, along track scanning, a layer of subswath formation would be natural, even if parallel processing control were not assigning subswath formation to separate nodes. Thus, when collection subswaths are present, their formation is what creates the first layer, iswl=1. A subswath layer is created after each step of coverage aggregation from

(after) the creation of scans to (before) the final full swath. So collecting scans into 4 sub swaths, collecting the 4 subswaths into 2 subswaths and collecting the 2 subswaths into the swath counts as 2 layers (one with 4 sub-swaths and one with 2 subswaths). This is consistent with the illustration in the definition of PF.Overlap.SubSwath.ID1(j).
If the number of layers is set to 0, it would imply that the collected scans were directly combined in one step to create the final swath. Either the entire collection swath consists of a single collection subswath, or no collection subswaths were used.
See the figure in PF.Overlap.SubSwath.ID1(j) for an example of the hierarchical accumulation of lower level subswaths into subswaths at the next level. In the case illustrated in that figure, PF.Subswath.Nlayers would have the value 3. The range of iswl in PF.Subswath.NumScans(iswl) values of *iswl* in PF.SubSwath.ScansList.*iswl*(isw,j) is from 1 to PF.Subswath.Nlayers.

Governing quality metrics: Must be correct value

See Also: PF.SubSwath.ScansList.*iswl*(isw,j)

20.3.54 PF.Subswath.NumScans(iswl) [L2]

Brief Definition: PF.Subswath.NumScans(iswl) gives the number of scans per subswath for each subswath layer.

indices: Index, iswl, ranges from 1 to PF.Subswath.Nlayers (or 0: (PF.Subswath.Nlayers-1)), that is over the subswath layers used to generate the current product.

Units/representation: Dimensionless, UInt32

Treatment/usage: Per the discussion of 10.1.4.3, a processing application is permitted to create subswaths by breaking the collected data up into groups of scans. The convention introduced here is that in any subswath layer, all of the subswaths must have the same number of scans. It is also expected that specific scans may be processed into separate subswaths at each level to create a region of subswath to subswath overlap that can be used to register subswaths to each other in a pairwise fashion. This restriction should cause minimal data wastage due to a last odd lot of scans. It follows from this rule that the number of scans per subswath is only a function of subswath layer number (iswl) and is not a function of subswath identifier, so no isw indices appear in PF.Subswath.NumScans(iswl).
This list includes collection subswath sizing, which will always appear at iswl=1 when active.

Governing quality metrics:

See Also: PF.Subswath.ScansList.*iswl*(isw,j)

20.3.55 PF.Subswath.NumSubswaths(iswl) [L2]

Brief Definition	PF.Subswath.NumSubswaths(iswl) gives the number of subswaths for each subswath layer.
indices	Index, iswl, ranges from 1 to PF.Subswath.Nlayers (or 0: (PF.Subswath.Nlayers-1)), that is over the subswath layers used to generate the current product.
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.1.4.3, a processing application is permitted to create subswaths by breaking the collected data up into groups of scans which can be collected into a number of subswaths to create a first layer of subswaths. That layer of subswaths can be aggregated into a smaller set of subswaths comprising a next layer, and so on until the final swath is ultimately formed. The number of subswaths present in each layer, indexed by iswl, is given by PF.Subswath.NumSubswaths(iswl). This determines the range of isw, the subswath index at each subswath layer.
Governing quality metrics	
See Also	PF.Subswath.ScansList.*iswl*(isw,j)

20.3.56 PF.Subswath.ScansList.*iswl*(isw,it,j) [L2,3,4,5]

Brief Definition	PF.Subswath.ScansList.*iswl*(isw,it,j) identifies which scans are included in a particular subswath as specified by a subswath layer index (iswl) and a subswath index for that layer (isw).
indices	Index, isw ranges from 1 to PF.Subswath.NumSubswaths(iswl) (or 0: (PF.Subswath.Nlayers-1)), that is over the subswaths at the level indicated by the value indicated by *iswl*. Index, it, ranges from PD.LoIndx.SubSwScGrpClasses(*iswl*) to PD.HiIndx.SubSwScGrpClasses(*iswl*), that is over the number of internally consecutive scan sequences (each of which has a single aspect (i.e., nadir-azimuth range for geometric aspect diversity distinct from that of the scan sequence before and/or after it) in the subswaths of layer *iswl*. Index,j, ranges from 1 to 5. The string *iswl* acts like an index, creating a separate parameter name for the scan identifier list defining a subswath for each subswath layer. This permits the use of an array for each level. A single array for all levels would have been awkward since the number of subswaths at each level varies.
Units/representation	Dimensionless, UInt32
Treatment/usage	Per the discussion of 10.1.4, subswaths are built up from scans. Subswaths are

constrained to be formed from integer scans. Thus, even subswaths at layer 2 or higher, explicitly formed by combining subswaths from a lower layer, can be defined in terms of which scans they contain. The main reason for tracking subswaths is to support the application of processing based corrections that may not have been applied in creating the fully processed point cloud as it appears in a given data set. The scan content of a subswath supports rapid determinination of the extent within the product that the processing block known as a subswath represents. This information, in turn, is needed to support determination of geolocation error bounds as it is needed to compute the weight for certain corrections arising out of internal registrations that may have been performed in processing the data.
PF.Subswath.ScansList.*iswl*(isw,it,j) constitutes a list of scans that are the content of each subswath. The sub- swath of interest is identified by a value of iswl and isw. The identities of the scans in a given subswath are documented via a list of "is" values, that is, the scan index counts for scans in a subswath at layer *iswl* is given.

This CMMD requires that scans in a subswath be *spatially* adjacent. For uniform progressive ground scanning, spatially contiguous scan blocks will also be sequential. So, in that case it is trivial to have the scan index range within a subswath be consecutive scan index values of "is" and the range of "it" would be degenerate. But when ground regions are revisited, as in modes designed to collect nadir diversity, spatial adjacency does not imply blanket temporal adjacency. So **temporally labeling scans (**as highly recommended in 10.1.4.2 **) with sequential values of "is" does not lead to** consecutive scan index values within a subswath in all cases. However, blocks of scans within any spatially contiguous subswath will be consecutive temporally, so we can completely specify the content of a spatial subswath with a list of several ranges of first and last scan index values. The "it" index enumerates the several ranges of first and last scan index values for temporal sequences of scans. If there are several subswath layers, the range of "it" will increase with the layers as the coverage of area by the subswaths increases.

PF.Subswath.ScansList.*iswl*(isw,it,1) is the pulse tile number (j_tile value) associated with defining the first scan[160] of a subswath's itth temporally consecutive scan sequence.
PF.Subswath.ScansList.*iswl*(isw,it,2)) is the scan identifier value of "is" as determined in SC.Pulse.Scan.Num.*TileNum*(ip,j) for the (temporally) first scan in the subswath's itth temporally consecutive scan sequence.

[160] SC.Pulse.Scan.Num.*TileNum*(ip, j) maps a pulse into a scan number.

PF.Subswath.ScansList.*iswl*(isw,it,3) is the pulse tile number (j_tile value) associated with defining the last scan of a sub- swath's itth temporally consecutive scan sequence.
PF.Subswath.ScansList.*iswl*(isw,it,4) is the tile specific scan identifier value of "is" as determined in SC.Pulse.Scan.Num.*TileNum*(ip,j) for the (temporally) last scan in the sub- swath's itth temporally consecutive scan sequence.
PF.Subswath.ScansList.*iswl*(isw,it,5) is the channel number[161] for the scans in the itth scan sequence contributing to the subswath specified by the subswath layer and subswath numbers *iswl* and isw respectively. In a single channel lidar, PF.Subswath.ScansList.*iswl*(isw,it,5) should be set to 1 as a default. In this case, we can expect that index, "it' will have a degenerate range as defined through PD.LoIndx.SubSwScGrpClasses(*iswl*) and PD.HiIndx.SubSwScGrpClasses(*iswl*).

Different subswath levels can have different numbers of scans. Low values of *iswl* will have fewer scans and as the subswaths roll up, they get larger. So we make iswl an evaluated part of the name of the ScansList giving the content list of scans in a subswath. Then, all subswaths at a given level have the same number of scans and the ranges of "it" and "j" are well defined.

Given a subswath identifying pair (iswl, isw), the constitutent scans can be looked up in PF.Subswath.ScansList.*iswl*(isw,it,j) . However, since subswaths are likely to be defined with spatial overlap, finding a scan in a single subswath defining list does not insure that one knows all of the subswaths that contain that scan. If a scan is near the boundary of a subswath, it may also be an element of the neighboring subswath. In a single subswath layer, iswl, a scan should appear in no more than 2 subswaths.

---

[161] In principle, a set of channels could share a spatial grouping. Suppose we have a dual band system that looks in two different nadir directions. Then there could be the forward short wavelength and forward long wavelength channel set and aft short wave and aft long wave channel set. Since both bands cover the same geography, only the spatial channelization needs to be identified for this purpose. However, we have so far only defined channel numbers, not channel groups. So if such a system were in use, we would pick only one channel number to carry the spatial info for the subswath definition. The subswath is fully defined in terms of collection timing and location error. The current metadata scheme works without explicit addition of the channel grouping.

Recall that channelization [vs treatment as separate sensors] implies correlation of location errors for separate channels by time of collection, not target position. So, conceivably, a revisited region could have the channel contributors to its subswath be decorrelated because the channels collect that region at sufficiently different times that the errors are no longer correlated. A processor would need to determine the need for channel registration in assembling subswaths of overlapping collection from times that are different enough to expect separate statistical draws of the geolocation error.

Governing quality metrics	
See Also	PF.Subswath.NumSubswaths(iswl) SC.Scan.ConePulseTable(j) SC.Pulse.Scan.Num.*TileNum*(ip,j) PD.HiIndx.SubScGrpClasses(*iswl*)

PF.Swath.*

PF.Swath.* parameters describe items processing items characteristic of the full swath product

20.3.57 PF.Swath.OffsetStatus [L2,3,4,5]

Brief Definition	PF.Swath.OffsetStatus indicates whether the supplied or applied geometric transformation data has multi-look or ground truth origin or is just from the single collection information.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	PF.Swath.OffsetStatus =0 means that any geometric transforms (6 parameter adjustments) supplied as metadata in this data set are due only to processing of data from the single radar collection processed to produce this data set. PF.Swath.OffsetStatus =1 means that the geometric transforms supplied as metadata in this data set include block adjustment based on a resection due to registration with other data sets to leverage available PDOP improvement on geometric accuracy. This value can be used to indicate that the data set is not level-4 but has the accuracy potential of the combination of two or more lidar collections with a good PDOP. This supports an anticipated strategy of not necessarily creating level-4 data sets with the extra wide band content, but leveraging the geolocation improvement of the multi-collect PDOP. PF.Swath.OffsetStatus =2 means that ground truth data was used to support computation of the geometric transforms to provide enhanced geometric accuracy PF.Swath.OffsetStatus =3 is like the meaning of setting this to 1, but that there is wide band data from multiple enterprise level-3s in the data set (so a value of 3 applies only to enterprise level-4 data sets) and the data from all sources is subject to separate PDOP based geometrical accuracy enhancement parameter sets. PF.Swath.OffsetStatus =4 means that post collection sensor platform trajectory information updates based on navigational system data but not lidar returns has been used. This could even be the case with enterprise level-1

data.

Note that there is no code for incorporating PDOP based block adjustments into the Tile Cartesian frame origins and orientations. This is because the corrections could be done at a subswath level and the entire data set could use a single Cartesian frame. We should not force separate Cartesian frames just to permit block adjustment. So we will always put the corrections into the transforms, not the frames. Those can be kept in ENU with or without block adjustments from multiple collection processing. If PDOP based block adjustments were applied by revising tile coordinate frame data, we would have crossed the line into level-3 data where we would not carry the adjustment data and would indicate the update via MD.ProcHist.ProdType.

Governing quality metrics

See Also PF.Adjustmts.Mean.Subswath(iswp, j) MD.ProcHist.ProdType

20.3.58 PF.Swath.SynBspread [L2,3,4,5]

Brief Definition PF.Swath.SynBspread is a characteristic value of the spectral **B**andwidth of synthetic points for the swath.

indices none

Units/representation Meters^{-1}, float

Treatment/usage This swath level value is derived from the distribution of values for PF.Point.SynBspread.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal ditstribution, the mode as chosen by the populating application to be a representative value. If it is known that synthetic point spectral width knowledge for individual points will be needed by downstream tools, the PF.Point.SynBspread.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynBspread[162], should be used when the possibility of a synthetic point having a distinctive enough spectral support to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of spectral support.

Governing quality metrics MAX(|PF.Swath.SynBspread-MAX(PF.Point.SynBspread.*TileNum*(ipt)|, |PF.Swath.SynBspread-MIN(PF.Point.SynBspread.*TileNum*(ipt)|) over the product

See Also

[162] To be clear, Swath versions of synthetic point spreads are to be used when a single value is characteristic of an entire data set and local variation can be ignored.

20.3.59 PF.Swath.SynPspread [L2,3,4,5]

Brief Definition	PF.Swath.SynPspread is a characteristic value of the **P**olarization variation contributing to synthetic points for the swath.
indices	none
Units/representation	degrees, float
Treatment/usage	This swath level value is derived from the distribution of values for PF.Point.SynPspread.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal ditstribution the mode as chosen by the populating application to be a representative value. If it is known that synthetic point polarization width knowledge for individual points will be needed by downstream tools, the PF.Point.SynPspread.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynPspread[162], should be used when the possibility of a synthetic point having a distinctive enough polarization support to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of polarization support.
Governing quality metrics	MAX(|PF.Swath.SynPspread-MAX(PF.Point.SynPspread.*TileNum*(ipt)|, |PF.Swath.SynPspread-MIN(PF.Point.SynPspread.*TileNum*(ipt)|) over the product
See Also	

20.3.60 PF.Swath.SynTasymtry [L2,3,4,5]

Brief Definition	PF.Swath.SynTasymtry is a characteristic value of the **T**ime asymmetry of synthetic points for the swath.
indices	none
Units/representation	Seconds, float
Treatment/usage	This swath level value is derived from the distribution of values for PF.Point.SynTasymtry.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal ditstribution the mode as chosen by the populating application to be a representative value. If it is known that synthetic point asymmetry knowledge for individual points will be needed by downstream tools, the PF.Point.SynTasymtry.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynTasymtry[162], should be used when the possibility of a synthetic point having a distinctive enough time asymmetry to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of time asymmetry.
Governing quality	MAX(|PF.Swath.SynTasymtry-MAX(PF.Point.SynTasymtry.*TileNum*(ipt)|,

metrics	|PF.Swath.SynTasymtry-MIN(PF.Point.SynTasymtry.*TileNum*(ipt)|) over the product
See Also	PF.Point.SynTasymtry.*TileNum*(ipt)

20.3.61 PF.Swath.SynTspread [L2,3,4,5]

Brief Definition	PF.Swath.SynTspread is a characteristic value of the contributing **T**ime interval of synthetic points for the swath.
indices	none
Units/representation	seconds, float
Treatment/usage	This swath level value is derived from the distribution of values for PF.Point.SynTspread.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal ditstribution the mode as chosen by the populating application to be a representative value. If it is known that synthetic point temporal width knowledge for individual points will be needed by downstream tools, the PF.Point.SynTspread.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynTspread, should be used when the possibility of a synthetic point having a distinctive enough spectral support to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of temporal support.
Governing quality metrics	MAX(|PF.Swath.SynTspread-MAX(PF.Point.SynTspread.*TileNum*(ipt)|, |PF.Swath.SynTspread-MIN(PF.Point.SynTspread.*TileNum*(ipt)|) over the product
See Also	

20.3.62 PF.Swath.SynXspread[L2,3,4,5]

Brief Definition	PF.Swath.SynXspread is a characteristic value of the x-direction support region of synthetic points for the swath.
indices	none
Units/representation	Meters, float
Treatment/usage	This swath level value is derived from the distribution of values for PF.Point.SynXspread.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal ditstribution the mode as chosen by the populating application to be a representative value. If it is known that synthetic point spatial width knowledge for individual points will be needed by downstream tools, the PF.Point.SynXspread.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynXspread[162], should be used when the possibility of a synthetic point having a distinctive enough

	spatial support to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of spectral support.
Governing quality metrics	MAX(|PF.Swath.SynXspread-MAX(PF.Point.SynXspread.*TileNum*(ipt)|, |PF.Swath.SynXspread-MIN(PF.Point.SynXspread.*TileNum*(ipt)|) over the product
See Also	

20.3.63 PF.Swath.SynYspread[L2,3,4,5]

Brief Definition	PF.Swath.SynYspread is a characteristic value of the y-direction support region of synthetic points for the swath.
indices	none
Units/representation	Meters, float
Treatment/usage	This swath level value is derived from the distribution of values for PF.Point.SynYspread.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal distribution the mode as chosen by the populating application to be a representative value. If it is known that synthetic point spatial width knowledge for individual points will be needed by downstream tools, the PF.Point.SynYspread.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynYspread, should be used when the possibility of a synthetic point having a distinctive enough spatial support to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of spectral support.
Governing quality metrics	MAX(|PF.Swath.SynYspread-MAX(PF.Point.SynYspread.*TileNum*(ipt)|, |PF.Swath.SynYspread-MIN(PF.Point.SynYspread.*TileNum*(ipt)|) over the product
See Also	

20.3.64 PF.Swath.SynZspread[L2,3,4,5]

Brief Definition	PF.Swath.SynZspread is a characteristic value of the z-direction support region of synthetic points for the swath.
indices	none
Units/representation	Meters, float
Treatment/usage	This swath level value is derived from the distribution of values for PF.Point.SynZspread.*TileNum*(ipt) over all of the points in the data set. It is given by a mean, median value, or in the case of a unimodal distribution the mode as chosen by the populating application to be a representative value. If it is known that synthetic point spatial width knowledge for individual points

	will be needed by downstream tools, the PF.Point.SynZspread.*TileNum*(ipt) should be populated. Those values, not PF.Swath.SynZspread, should be used when the possibility of a synthetic point having a distinctive enough spatial support to lead to a location uncertainty bias of concern exists. When using the characteristic value over the swath is benign relative to using a point specific value, data creating or exploitation applications may decide not to populate/use the point specific values of spectral support.
Governing quality metrics	MAX(|PF.Swath.SynZspread-MAX(PF.Point.SynZspread.*TileNum*(ipt)|, |PF.Swath.SynZspread-MIN(PF.Point.SynZspread.*TileNum*(ipt)|) over the product
See Also	

PF.SynMethod.*

PF.SynMethod.* parameters describe items documenting processing methods to arrive at synthetic lidar point elements position, time and intensity.

20.3.65 PF.SynMethod.Intensity [L2,3,4,5]

Brief Definition	PF.SynMethod.Intensity is a code describing the method of computing the intensity for synthetic lidar points in the data set.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	A synthetic lidar point associates values of statistics of lidar data with a particular voxel in the scene region. One such statistic is the intensity of the synthetic lidar point. The following list defines the values that may be assigned to the code for synthetic intensity method. Code for method to obtain resultant intensity: 1--Voxel maximum. When Multiple lidar points are present within a voxel, the maximum intensity of any of the points becomes the voxel value. This would not be a viable choice for Geiger mode data since the lidar points in that mode have no native intensity. 2--Voxel only intensity wtd averaging. When multiple lidar points are present in a voxcel, a weighted average of the intensities becomes associated with the voxel. The intensity based weights for this method are not specified but could be, for example, based on the square of measured photo current at a lidar point. 3--voxel nearest center value. This value applies if the voxel carries the intensity of the nearest point to the voxel center. This would not be a viable choice for Geiger mode data since the lidar points in that mode have no native intensity. 4--Voxel median. This value applies if the value is that of the median lidar

point in the voxcel. If there are an even number of points, the median will be taken as the mean of the two mid distribution points
5--Voxel centroid, moment wtd. In this method, a weighted average is also used, but rather than using intensity as the basis for computing weights, only the distance of the lidar points from the voxel center is used to arrive at a weight.
6— Voxel centroid, Mixed weighting depends on both intensity and lidar point position.
7—percent of possible GM hits[163]. This is quite appropriate for Geiger mode. This value indicates that intensities are reported as the percentage of interrogations of the voxel with actual photon return. So, this really is a local point density being simplistically interpreted as scene reflectivity via reporting synthetic point "intensity". Since this depends on voxel size in each dimension, this choice should be complemented by explicit population of PF.Swath.SynXspread or PF.Point.SynXspread.*TileNum*(ipt) etc. , which give the effective sizes of spatial increments in voxel space for creating synthetic points.
8—pseudo reflectivity. For GM, this is the percent of possible GM hits for a voxel where the voxel interrogation count is reduced by the number of pulse-rays intersecting the voxel that have returns from ranges closer than the voxel but within the quench gate[164] of the voxel. This tends to avoid the "darkening" of near ground voxels due to extinction from volume scattering and noise in the sensor quench gate. For LM, the analogous correction for beam depletion is applied to make the intensity values in LD.Point.Intensity.*TileNum*(ipt,j) model voxel reflectivity.
9—Synthetic point intensity is determined by deconvolution of the point cloud with an estimated or a'priori point spread function.

Note that code values of 7 and 8 for PF.SynMethod.Intensity would only be used for Geiger mode data processing. The population of PF.SynMethod.Intensity with either value needs to be consistent with information provided in the user definition field, PD.UserDefined.PtDef.P*n* for UD.PtDat.P*n*.*TileNum*(ipt). PF.SynMethod.Intensity in this case need not be populated except to provide a convenient high level characterization of

[163] This is not intended to describe the data in LD.Point.Intensity.*TileNum*(ipt,j). See LD.Point.Intensity.*TileNum*(ipt,j) for discussion of user defined reflectivity that would be populated with a code 8 here and a specific formula in the user defined tag section.
[164] For a single quench GM mode, the whole pulse receive window is the quench gate, so any return on the ray "before" the voxel would reduce the interrogation (pulse -ray) count for the voxel.

PD.UserDefined.PtDef.P*n*.

Governing quality metrics

See Also ES.Scale.X_Fac LD.Point.Intensity.*TileNum*(ipt) PF.Point.SynXspread.*TileNum*(ipt) PF.Swath.SynXspread

20.3.66 PF.SynMethod.Position [L2,3,4,5]

Brief Definition PF.SynMethod.Position is a code describing the method of computing the position for synthetic lidar points in the data set.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage A variety of methods are conceivable for how synthetic point positions might be generated from measured lidar point positions. The code values for the method to obtain resultant position are:

1--Voxel centroid,uniform average. In this method, an accumulation voxel that is sized large enough to assure a large number of lidar points in the vicinity of real returns is used. The reporting voxel grid, as documented in ES.Scale.X_Fac, ES.Scale.Y_Fac, ES.Scale.Z_Fac, is considerably smaller than the accumulation voxel. The reported synthetic XYZ for the accumulation volume based a local statistic is a uniform average of measured point position, truncated to the reporting grid precision.

2--Voxel centroid,intensity wtd average. In this method, an average is formed similarly to the description for the code value of 1. The difference is that in this method, the centroid is computed via an intensity weighted average. This is only suitable when the lidar points have native intensity values, as for example the case of linear mode operation.

3--voxel center This method is quite simplistic. The accumulation voxel is by construction centered on the reporting grid position for every output point.

4--Voxel centroid, moment wtd. This method is analogous to the second method, but uses distance from the accumulation center as the basis for weighting.

5—Accumulation voxel centroid is computed using weights derived from both intensity and distance from the accumulation center.

6—median The reported XYZ positions are the medians of the lidar points in the accumulation voxels.

7—Synthetic point position is determined by deconvolution of the point cloud with an estimated or a'priori point spread function.

Governing quality metrics

See Also ES.Scale.X_Fac

20.3.67 PF.SynMethod.Time[L2,3,4,5]

Brief Definition	PF.SynMethod.Time supplies a code to describe how time is determined for synthetic lidar points in the current data set.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	Synthetic lidar points are defined in section 5.1. To support full exploitation capability, including geo-position error propagation and obscuration related effects analysis, a time tag needs to be associated with every synthetic lidar point so that it has the properties of a fully documented (natural) lidar point. Since the process of creating a synthetic lidar point is one of creating a statistic from a collection of lidar points, each with potentially different times of collection, there is a degree of arbitrariness that is associated with assigning a time to a synthetic lidar point. Insight into the physical impacts of the applied time tagging process are available through the values of PF.SynMethod.Time. The codes for the method to obtain the time tag for synthetic points are: PF.SynMethod.Time =1—uniformly weighted average of contributing point times PF.SynMethod.Time =2--intensity wtd average of contributing point times PF.SynMethod.Time =3--moment wtd average of contributing point times PF.SynMethod.Time =4--mode, that is, the most frequent value of contributing point times PF.SynMethod.Time =5—median, the middle time value in the list of contributing point times (even numbered list can average the two mid list times) PF.SynMethod.Time =6-- Time of natural point with closest value of XYZ to the synthetic lidar point position …
Governing quality metrics	
See Also	PF.SynMethod.Position PF.Swath.SynTspread PF.Swath.SynTasymtry

PF.TimeToRange.*

PF.TimeToRange.* parameters describe items documenting the parameters and algorithm for converting photon travel time to range and XYZ position

20.3.68 PF.TimeToRange.BendEffect [L2,3,4,5]

Brief Definition	PF.TimeToRange.BendEffect is a code for what refractive impacts have been included in XYZ calculation.
indices	none

Units/representation	Dimensionless, UInt8
Treatment/usage	The physical impacts of light refraction along the ray path include slower than vacuum propagation speed, ray bending, atmospheric piston Doppler, scintillation and dispersion. For volume return products and certain collection conditions, the impacts may have insignificant magnitude (relative to applicable accuracy requirements) but add significant modeling costs due to engineering complexity or run time. The processor application will determine what effects to include in the (ray, range) to (X, Y, Z) conversion. PF.TimeToRange.BendEffect passes this information on to users. Code values are: PF.TimeToRange.BendEffect=0 – Propagation treated as if in a vacuum PF.TimeToRange.BendEffect=1-- only delay but accounting for optical path dilation with curvature, XYZ from straight rays PF.TimeToRange.BendEffect=2-- Delay and XYZ from curved rays. deviation due to apparent elevation angle not equal to straight ray approximation. …
Governing quality metrics	
See Also	

20.3.69 PF.TimeToRange.Model [L2,3,4,5]

Brief Definition	PF.TimeToRange.Model carries a code for the model used for converting photon time of flight to range.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	Various models have solved the ray/delay problem to different levels of accuracy. The earliest efforts were limited in wavelength and assumed full atmosphere propagation paths as if the platform were in space. Subseqently, different approximations have arrived that permit the ray paths to traverse only part of the atmosphere, model the atmosphere more closely, etc. Knowledge of the time to range algorithm gives insight to the credibility of claimed range uncertainty data. The Processor may document the time-to-range algorithm by name with the following codes: PF.TimeToRange.Model =1 --Marini Murray PF.TimeToRange.Model =2 --Revised Marini Murray PF.TimeToRange.Model =3 --Yan Wang PF.TimeToRange.Model =4 --Hulley Pavlis PF.TimeToRange.Model =5 --IROA Integral Ray Optics Approximation PF.TimeToRange.Model =6 –Gyer 1996 [Photogrammetric Engineering & Remote Sensing, Vol. 62, No. 3, March 1996, pp. 301-310.]

PF.TimeToRange.Model =7 --Unpublished vendor proprietary
…

Governing quality metrics

See Also

20.3.70 PF.TimeToRange.Parms.Ht_Surface(j) [L1,2]

Brief Definition: PF.TimeToRange.Parms.Ht_Surface(j) gives the value of scene base height to support atmospheric model surface gravity determination in the time to range conversion model.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Units/representation: Meters, Float

Treatment/usage: Atmospheric models for refraction may include the impact of the weight of the atmospheric column. Typically this is derived from the scene base height above the ellipsoid. Clearly, if the lidar is being used to find the actual scene heights, this parameter can only be approximately known in advance of processing the lidar data. PF.TimeToRange.Parms.Ht_Surface(j) provides the value used in determining the atmospheric propagation correction in the processor. Caution, published equations may expect units of kilometers, so mind the units.

Governing quality metrics

See Also

20.3.71 PF.TimeToRange.Parms.Hu_Surface(j) [L1,2]

Brief Definition: PF.TimeToRange.Parms.Hu_Surface(j) gives the surface value of humidity in terms of water partial pressure.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.

Units/representation: Millibars, Float

Treatment/usage: The deviation of the speed of light in the atmosphere from that in vacuum is significantly influenced by the humidity. The humidity is thus a common model parameter for atmopspheric refraction.
PF.TimeToRange.Parms.Hu_Surface(j) provides the surface Humidity (partial pressure of water millibars) assumed in performing the time to range conversion for the current data set.

Governing quality metrics

See Also

20.3.72 PF.TimeToRange.Parms.Lat(j) [L1,2]

Brief Definition	PF.TimeTo Range.Parms.Lat(j) gives the latitude of the scene for the current data set that was used in the refraction calculation.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Units/representation	Degrees, Float
Treatment/usage	In determing the weight of the atmospheric column, not only does the scene base height contribute, but the problem is latitude dependent due to ellipsoid/geoid effects and the Earth's rotation. Some detailed models will need the scene latitude to be input to predict refractive delay. PF.TimeTo Range.Parms.Lat(j) provides the value used in the refraction calculation.
Governing quality metrics	
See Also	

20.3.73 PF.TimeToRange.Parms.Nsurface(j) [L1,2]

Brief Definition	PF.TimeToRange.Parms.Nsurface(j) provides the index of refraction at the scene surface.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Units/representation	Dimensionless, Float
Treatment/usage	The refractive index of air is close to unity. To maximize the value of the precision of this parameter, PF.TimeToRange.Parms.Nsurface(j) is only the fractional part of the index of refraction, 1.000nnnn . The number 0.000nnnn [that is, the index of refraction at the scene surface minus 1.0000000] expressed as a floating point number is stored in PF.TimeToRange.Parms.Nsurface(j) when reporting the value of surface index of refraction used as an input to the refraction model.
Governing quality metrics	
See Also	

20.3.74 PF.TimeToRange.Parms.P_Surface(j) [L1,2]

Brief Definition	PF.TimeToRange.Parms.P_Surface(j) gives the surface atmospheric pressure used in the time to range conversion.
indices	Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-processing position and LOS uncertainty.
Units/representation	Millibars, float
Treatment/usage	The refractive index of air depends on the pressure. The atmospheric

refraction model may either use the surface air pressure as an input or produce it from other parameters. PF.TimeToRange.Parms.P_Surface(j) will be populated in enterprise L-1 and L-2 data sets with a positive value when the number represents a model input, and will be given a negative value in L-2 data sets when PF.TimeToRange.Parms.P_Surface(j) reports a value derived by the model from other inputs. In the case where the surface pressure is derived, the surface pressure is not an input parameter so, PF.TimeToRange.Parms.P_Surface(j) would not be populated in enterprise L-1 data sets and would be reported in L-2 data sets with the minus sign flag.

Governing quality metrics

See Also

20.3.75 PF.TimeToRange.Parms.ScaleHeight(j) [L1,2]

Brief Definition: PF.TimeToRange.Parms.ScaleHeight(j) carries an array of scale heights relevant to stratified atmospheric models that may be invoked to determine the time to range conversion.

indices: Index, j=1,2,3,4

Units/representation: Meters, float

Treatment/usage: We permit carrying 4 scale heights for use in the atmospheric refraction model. When none of these are actually used, the entire parameter need not be populated. When at least one of the following slots is in use, the entire array should be used, with the value -1.0000E+00 used to indicate an unused element

PF.TimeToRange.Parms.ScaleHeight(1) = scale height for pressure
PF.TimeToRange.Parms.ScaleHeight(2) = scale height for temperature
PF.TimeToRange.Parms.ScaleHeight(3) = scale height for humidity
PF.TimeToRange.Parms.ScaleHeight(4) = scale height for refractive index

A scale height determines the rate of exponential decay of a quantity through a function of the form, exp[-z/H] where z is the height above a reference surface and H is the scale height. The details of the scale height definition are as assumed by the algorithm named in PF.TimeToRange.Model.

Governing quality metrics

See Also: PF.TimeToRange.Model

20.3.76 PF.TimeToRange.Parms.T_Surface(j) [L1,2]

Brief Definition: PF.TimeToRange.Parms.T_Surface(j) gives the surface temperature used in the atmospheric refraction model for time to range conversion.

indices: Index j, ranges from PD.LoIndx.Scan(1) to PD.HiIndx.Scan(1), generally from 1 to the number of snapshots of the error covariance matrix for pre-

	processing position and LOS uncertainty.
Units/representation	Kelvins, float
Treatment/usage	Atmospheric models are dependent on the absolute temperature. PF.TimeToRange.Parms.T_Surface(j) gives the surface temperature used in the time to range conversion.
Governing quality metrics	
See Also	

PF.UserDefined.*

PF.UserDefined.* parameters describe items documenting the parameters defined in the metadata by User/Developers and carried by data buckets provided for in this CMMD. When a user defined parameter becomes accepted, it should be given a descriptive name and added to the CMMD.

20.3.77 PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) [L1,2,3,4,5]

Brief Definition	PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) carries narrow band user defined data at the tile level.
indices	Index, j, ranges from PD.LoIndx.UDNB*n* to PD.HiIndx.UDNB*n* The meaning of the range of j for a given value of n is to be included in the explanatory text in PD.UserDefined.NB_Def.P*n*. *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. The evaluated sub-string "*n*" also behaves somewhat as an index. A separate value of *n* is used for each user defined narrow band parameter. So, in the case where a user has definitions for a first and second narrow band parameter, the data set would contain two narrow band data items. These might be, for example, PF.UserDefined.NB_Dat.P1.*TileNum*(j) and PF.UserDefined.NB_Dat.P2.*TileNum*(j).
Units/representation	Variable, see PD.UserDefined.NB_Type.P*n*
Treatment/usage	PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) is one of 3 groups of data fields for user defined data. Each value of the evaluated string, *n*, indicates a distinct user defined narrow band data parameter. To support the possibility that a user defined narrow band data item has a unique value for different tiles, the *TileNum* evaluated string is supplied. Each PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) item, with evaluated *n* field, is given its actual definition in the text string which is the value of

PD.UserDefined.NB_Def.P*n*. The purpose of providing such user defined data fields is to provide flexibility to special user needs and to avoid the problem of minor variations in weighting, terms, filter sizes leading to a proliferation of entities/tags or compromise interoperability via near miss definition issues.

Governing quality metrics

See Also PD.UserDefined.NB_Type.P*n* PD.UserDefined.NB_Def.P*n* PD.HiIndx.UDNB*n*

20.4 PD.* Product Definition

PD.DatDscrptors.*

PD.DatDscrptors.* data items are data descripttors that specify values for generic conceptual data that need to be customizable for various implementations

20.4.1 PD.DatDscrptors.Uncerty(j) [L1,2,3,4,5]

Brief Definition PD.DatDscrptors.Uncerty(j) provides the percentile values for high confidence uncertainty statistics that can be generated using provided standard deviation and conversion factors.

indices Index, j=1,2

Units/representation Dimensionless, single float

Treatment/usage There are two distinct needs with respect to statistical uncertainty that are supported by this CMMD:

- Standard deviation/ variance data for rigorous error propagation and other probablility calculations
- High confidence (e.g. 90th percentile) error bar information

Since not all error distributions are Gaussian, this CMMD adopts a more general approach that permits data providers to use their knowledge of applicable error distributions to convert standard deviation or variance values of error to percentile errors via multiplicative factors provided as metadata in ES.Swath.HighConfdncP_Fac(i,j,k). The intent of the high confidence uncertainty data is to support users in regulating the occurrence rate of blunders due to relatively infrequent events. However, depending on the application, the degree of conservatism required is know to vary. Consequently we provide flexibility in the metadata support for high confidence uncertatnty documentation in two ways:

- We allow populating two[165] different values of the high confidence metrics in each dataset.

[165] As of writing of version 1.0, it is known that there is a need for at least, and most likely just, two levels at a time. Thus, to promote simplicity, we have hardwired the number of such metrics that can be populated in a single dataset

- We allow directly defining the value of the high confidence levels in each dataset. For some situations, .9P and .95P are the likely values. In other cases, perhaps .99P and .85P might be desired.

The choices of the high confidence percentiles are given in PD.DatDscrptors.Uncerty(j) and the conversion factors that multiply standard deviations to give the parameter values corresponding to these high confidence percentiles are given in ES.Swath.HiConfdncP_Fac(i,j,k).

Governing quality metrics

See Also ES.Swath.HiConfdncP_Fac(i,j,k)

PD.Hi/LoIndx.*

PD.Hi/LoIndx.* data items define index ranges for arrays. This supports transport independent data definition.

20.4.2 PD.HiIndx.AdjacentP(j) [L2,3,4,5]

Brief Definition: PD.HiIndx.AdjacentP(j) provides the ending values for the TileMap point tile adjacency metadata.

indices: Index, j=1,2

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.HiIndx.AdjacentP(j) supports the use of tile adjacency information carried in PD.TileMap.AdjacencyPts(j). PD.TileMap.AdjacencyPts(j) is unusual within this CMMD in that the range of its index, j, is j=1,2 but the range of the index for companion metadata, PD.LoIndx.AdjacentP(j), is j=1,2,3.

PD.HiIndx.AdjacentP(1) is the ending index value for the n_tile list.
If PD.LoIndx.AdjacentP(1) =1, then
PD.HiIndx.AdjacentP(1) = Number of tiles in the spatial tile adjacency map

PD.HiIndx.AdjacentP(2) is the ending index value for the column (second, i.e., j) index of the tile adjacency map.
If PD.LoIndx.AdjacentP(2) =1, then
PD.HiIndx.AdjacentP(2) = Number of tiles in a row of the spatial tile adjacency map

---

to two. If in the future it becomes necessary to permit an arbitrary number of such metrics, PD.Lo/HiIndx support for a variable length list can be instituted in a backwards compatible manner {by defaulting to 2} in later versions.

If PD.LoIndx.AdjacentP(3) =1 also, then
The number of rows in the tile map is
[PD.HiIndx.AdjacentP(1)/ PD.HiIndx.AdjacentP(2)]
The index of the last row of the tile adjacency map is the number of rows minus 1 plus PD.LoIndx.AdjacentP(3).

Default when not populated:
If PD.HiIndx.AdjacentP(j) is not populated, no adjacency information for point tiles is supported and using applications should not attempt to use any values for PD.HiIndx.AdjacentP(j).

Governing quality metrics

See Also PD.TileMap.AdjacencyPts(j)

20.4.3 PD.LoIndx.AdjacentP(j) [L2,3,4,5]

Brief Definition PD.LoIndx.AdjacentP(j) provides the starting values for the TileMap point tile adjacency metadata.

indices Index, j=1,2,3

Units/representation Dimensionless, UInt8

Treatment/usage PD.LoIndx.AdjacentP(j) supports use of 2-D tile adjacency information carried in PD.TileMap.AdjacencyPts(j).

PD.LoIndx.AdjacentP(1) is the index of the first tile in the point tile spatial adjacency map {lowest value of n_tile}

PD.LoIndx.AdjacentP(2) is the index of first column in the point tile spatial adjacency map

PD.LoIndx.AdjacentP(3) is the index of first row in point tile spatial adjacency map

Default when not populated:
If PD.LoIndx.AdjacentP(j) is not populated, no adjacency information for point tiles is supported and using applications should not attempt to use any values for PD.LoIndx.AdjacentP(j).

Governing quality metrics

See Also PD.HiIndx.AdjacentP(j)

20.4.4 PD.HiIndx.AdjacentW(j) [L1,2,3,4,5]

Brief Definition PD.HiIndx.AdjacentW(j) provides the ending values for the TileMap

	waveform tile adjacency metadata.
indices	Index, j=1,2
Units/representation	Dimensionless, UInt8

Treatment/usage	PD.HiIndx.AdjacentW(1) is the ending index value for the n_tile list.
	If PD.LoIndx.AdjacentW(1) =1, then
	PD.HiIndx.AdjacentW(1) = Number of tiles in the waveform spatial adjacency tile map
	
	PD.HiIndx.AdjacentW(2) is the ending index value for the column (second, i.e., j) index of the tile map
	If PD.LoIndx.AdjacentW(2) =1, then
	PD.HiIndx.AdjacentW(2) = Number of tiles in a row of the waveform spatial adjacency tile map
	
	If PD.LoIndx.AdjacentW(3) =1 also, then
	The number of rows in the waveform spatial adjacency tile map is [PD.HiIndx.AdjacentW(1)/ PD.HiIndx.AdjacentW(2)]
	The index of the last row of the tile map is the number of rows minus 1 plus PD.LoIndx.AdjacentW(3).
	Note that there is no PD.HiIndx.AdjacentW(3), despite the existence of PD.LoIndx.AdjacentW(3). This is due to dependence only the value of the first row's index in this scheme.
	
	<u>Default when not populated:</u>
	If PD.HiIndx.AdjacentW(j) is not populated, no adjacency information for waveform tiles is supported and using applications should not attempt to use any values for PD.HiIndx.AdjacentW(j).

Governing quality metrics	
See Also	PD.TileMap.AdjacencyWFs(j)

20.4.5 PD.LoIndx.AdjacentW(j) [L1,2,3,4,5]

Brief Definition	PD.LoIndx.AdjacentW(j) provides the starting values for the TileMap waveform tile adjacency metadata.
indices	Index, j=1,2,3
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.LoIndx.AdjacentW(j) supports use of 2-D tile adjacency information carried in PD.TileMap.AdjacencyWFs(j). PD.LoIndx.AdjacentW(1) is the index of the first tile in the waveform tile spatial adjacency map {lowest value of n_tile}

PD.LoIndx.AdjacentW(2) is the index of first column in the waveform tile spatial adjacency map

PD.LoIndx.AdjacentW(3) is the index of first row in waveform tile spatial adjacency map

Default when not populated:
If PD.LoIndx.AdjacentW(j) is not populated, no adjacency information for waveform tiles is supported and using applications should not attempt to use any values for PD.LoIndx.AdjacentW(j).

Governing quality metrics

See Also

20.4.6 PD.HiIndx.Array.Elem(ia,j) [L1,2]

Brief Definition

PD.HiIndx.Array.Elem(ia,j) provides the ending values of the total and bad element index ranges for the iath array.

indices

Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is for example, from 1 to the number of arrays.
Index, j, =1 for all elements in array, ia
Index, j, =2 for the bad elements only in array, ia
Index, j, =3 for the rows in the cross talk data array
Index, j, =4 for the columns in the cross talk data array

Units/representation

Dimensionless, UInt8

Treatment/usage

This data item describes the indexing for arrays, such as SC.Array.RayDirs.*Ia*(icase,ie,j), that use "ie" as an argument. If the first element index of focal plane array, ia, has the value, 1, PD.HiIndx.Array.Elem(ia,1), which is the last element index of array, ia, equals the number of elements in that array. This metadata item is separately defined to assure that a transport independent scheme for keeping track of the index range of focal plane array elements can be supported. This CMMD uses a general approach, which is robust to the possibility of sparse block population of data arrays. This permits the use of a single method across conceptual models that also supports data sets which might provide values for only part of a data array in a single named variable.
The number of elements is provided directly by SC.Array.LatticeDesc(ia,1), but that item does not define the populated index range. Certain implementations may choose to base array indexing at 0. The physical association of hardware detector elements to the element index sequence, ie, must be consistent throughout any single data set.

If PD.LoIndx.Array.Elem(ia,2) is set to 1, PD.HiIndx.Array.Elem(ia,2) is the

number of bad elements in array, ia, and their locations are provided by SC.Array.BadElmnts.*Ia*(j,k) for k=1 and 2.
If PD.LoIndx.Array.Elem(ia,3) is set to 1, PD.HiIndx.Array.Elem(ia,3) is the number of rows in the cross talk data for array, ia, which is provided by SC.Array.CrossTalkQ1.*Ia*(j,k,n).
If PD.LoIndx.Array.Elem(ia,4) is set to 1, PD.HiIndx.Array.Elem(ia,4) is the number of columns in the cross talk data for array, ia, which is provided by SC.Array.CrossTalkQ1.*Ia*(j,k,n).

Default when not populated:
If PD.HiIndx.Array.Elem(ia,j)is not populated, no array element data is given and PD.HiIndx.Array.Elem(ia,j) should not be used by a using program.

Governing quality metrics

See Also: SC.Array.RayDirs.*Ia*(icase,ie,j) PD.HiIndx.Array(1) SC.Array.BadElmnts.*Ia*(j,k) SC.Array.CrossTalkQ1.*Ia*(j,k,n)

20.4.7 PD.LoIndx.Array.Elem(ia,j) [L1,2]

Brief Definition: PD.LoIndx.Array.Elem(ia,j) is the starting value of the element indices for the iath array.

indices: Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is for example, from 1 to the number of arrays.
Index, j, =1 for all elements in array, ia
Index, j, =2 for the bad elements in array, ia
Index, j, =3 for the rows in the cross talk data array
Index, j, =4 for the columns in the cross talk data array

Units/representation: Dimensionless, UInt8

Treatment/usage: If PD.LoIndx.Array.Elem(ia,1) has the value, 1, PD.HiIndx.Array.Elem(ia,1), the last populated element index of array, ia, equals the number of elements in that array. This metadata item is separately defined to assure that a transport independent scheme for keeping track of the index range of focal plane array elements can be supported. This CMMD uses a general approach, which is robust to the possibility of sparse block population of data arrays. This permits the use of a single method across conceptual models that also supports data sets which might provide values for only part of a data array in a single named variable.
The number of elements is provided directly by SC.Array.LatticeDesc(ia,1), but that item does not define the populated index range. Certain implementations may choose to base array indexing at 0. The physical association of hardware detector elements to the element index sequence, ie, must be consistent throughout any single data set.
If PD.LoIndx.Array.Elem(ia,2) is the starting value of j for the bad element

list for array, ia, which is provided by SC.Array.BadElmnts.*Ia*(j,k). It will nominally be set to 0 or 1, depending on the indexing convention of the writing application.
If PD.LoIndx.Array.Elem(ia,3) is the starting value of j for the rows of the cross talk data for array, ia, which is provided by SC.Array.CrossTalk.*Ia*(j,k,n). It will nominally be set to 0 or 1, depending on the indexing convention of the writing application. This a case where there may be a convenience factor in making the first index value be 0.
If PD.LoIndx.Array.Elem(ia,4) is the starting value of k for the columns of cross talk data for array, ia, which is provided by SC.Array.CrossTalk.*Ia*(j,k). It will nominally be set to 0 or 1, depending on the indexing convention of the writing application. This a case where there may be a convenience factor in making the first index value be 0.

Default when not populated:
If PD.LoIndx.Array.Elem(ia,j)is not populated, no array element data is given and PD.HiIndx.Array.Elem(ia,j) should not be used by a using program.

Governing quality metrics

See Also SC.Array.BadElmnts.*Ia*(j,k)

20.4.8 PD.HiIndx.Array(j) [L1,2]

Brief Definition PD.HiIndx.Array(j) gives the ending index values for SC.Array.* and related items.

indices Index, j, ranges from 1 to 12, the number of upper index range values needed to describe the focal plane array support data.

Units/representation Dimensionless, Int8

Treatment/usage PD.HiIndx.Array(j) for
j=1, is the ending value of the index, ia, that is, the index of the last array in the sensor. For sensors with multiple focal plane arrays, the ordering must be maintained consistently for spectral, polar, etc. data. PD.HiIndx.Array(1) applies to data arrays using “ia” as an index.

j=2, is the ending value of the first index, ifc, of the (spectral) Filter data [SC.Array.FilterDat(ifc,j)]; it determines the number of specified optical band pass filters.

j=3, is the ending value of the index, j, and specifies the (spectral) Filter data SC.Array.FilterDat(ifc,j) second index range. The current value[166] is PD.LoIndx.Array(3) + 8.
This determines the number of specified optical band pass filter parameters.

j=4, is the ending value of the index, ipc, of (polarization) filter data [SC.Array.PolDat(ipc,j)] first index , and determines the number of specified polarizers

j=5, is the ending value of the index, j, of (polarization) filter data second index, determines the number of specified polarizer parameters. The current value[166] is PD.LoIndx.Array(5) + 58.

j=6, is the ending value of the index, icase, of element look direction data first index. This indicates the number of sets of values for the look directions. Element look directions may vary depending on the direction of the optical axis.
j=7, is the ending value of the second index, j, of SC.Array.LatticeDesc(ia,j), the lattice descriptor list
j=8, is the ending value of global index, iray [SC.WF.RayDesc.ArryGlbl(iray)]
j=9, is the ending value of index, j, for SC.WF.TxProfile.Flags(j)
j=10, is the ending value of index, j, for SC.Array.Chunks(ia,ich,j) ranges over chunk descriptors, currently set at 6.
j=11, is the ending value of the index that ranges over the VQ code table entries- governs icode in SC.WF.VQ_Decode(ia,icode).
j=12, is the ending value of the index that ranges over the waveform clarity parameters, currently supports 7 defined parameters in SC.Array.CurrentResp.Dat(ia,j)

<u>Default when not populated:</u>
If PD.HiIndx.Array(j) is not populated, it means that the data set cannot support documentation of array characteristics. No use of metadata supported capabilities with respect to multiple arrays, spectral or polarization filter specification, gimbal dependent array element look directions, focal plane detector layout, vector quantization of modulated waveform data, or the response of detector output to incident photon flux can be successful.

Governing quality metrics

[166] This value must be maintained as the number of spectral filter parameters evolves in later editions of this CMMD.

See Also PD.TileMap.WF_Returns(ia,j), SC.Array.PolCode(ia), SC.Array.RayDirs.*Ia*(icase, ie, j), SC.Array.RayDirsSigmas(ia,icase,j), SC.WF.RayDesc.ArryGlbl(iray), SC.Array.LatticeDesc(ia,j), SC.Swath.RateFactor(ia), SC.WF.Rx.Nsamps(ia), SC.WF.TxProfile.FM_Slope(ia), SC.WF.TxProfile.FM_Start(ia), SC.WF.TxProfile.FM_PhaseStart(ia), SC.WF.TxProfile.Dat(ia,j), SC.WF.TxProfile.Flags(j), SC.WF.TxProfile.Peak(ia), SC.WF.TxProfile.Nsamps(ia), SC.WF.VQ_Decode(ia,icode), SC.Array.FilterDat(ifc,j), SC.Array.PolDat(ipc,j), SC.Array.RayDirs.*Ia*(icase,ie,j) SC.Array.CurrentResp.Dat(ia,j)

20.4.9 PD.LoIndx.Array (j) [L1,2]

Brief Definition PD.LoIndx.Array(j) gives the starting index values for data associated with the focal plane arrays

indices Index, j, ranges from 1 to 12, the number of lower index range values needed to describe the focal plane array support data.

Units/representation Dimensionless, Int8

Treatment/usage PD.LoIndx.Array(j) for

j=1, is the starting value of the index, ia, that is, the index of the first array[167] in the sensor. For sensors with multiple focal plane arrays, the ordering must be maintained consistently for spectral, polar, etc. data. PD.LoIndx.Array(1) applies to data arrays using "ia" as an index.

j=2, is the starting value of the first index, ifc, of the (spectral) Filter data [SC.Array.FilterDat(ifc,j)], and determines the number of specified optical band pass filters

j=3, is the starting value of the index, j, the (spectral) Filter data second index [see SC.Array.FilterDat(ifc,j)]. Determines the number of specified optical band pass filter parameters.

j=4, is the starting value of the index, "ipc", the (polarization) filter data [SC.Array.PolDat(ipc,j)] first index , and determines the number of specified polarizers

j=5, is the starting value of the index, j, the (polarization) filter data second index, which determines the number of specified polarizer parameters

j=6, is the starting value of the index, "icase", of element look direction data first index. This indicates the number of sets of values for the look directions. Element look directions may vary depending on the direction of the optical axis.

j=7, is the starting value of the second index, j, of SC.Array.LatticeDesc(ia,j),

[167] Remember, a single pixel flying spot scanner uses a 1 x 1 array!

the lattice descriptor list
j=8, is the starting value of global index, iray [SC.WF.RayDesc.ArryGlbl(iray)]
j=9, is the starting value index, j, for the TxProfile Flags [SC.WF.TxProfile.Flags(j)]
j=10, is the starting value of index, j, for SC.Array.Chunks(ia,ich,j) when this is set to 1, PD.HiIndx.Array(10) will currently be set to 5.
j=11, is the starting value of the index that ranges over the VQ code table entries, governs icode in SC.WF.VQ_Decode(ia,icode).
j=12, is the starting value of the index (second index,j) that ranges over the waveform clarity parameters in SC.Array.CurrentResp.Dat(ia,j).

The data set writing code may choose to start array indexing at 0 or 1 and use this metadata item to document its choice.

Default when not populated:
If PD.LoIndx.Array(j) is not populated, it means that the data set cannot support documentation of array characteristics. No use of metadata supported capabilities with respect to multiple arrays, spectral or polarization filter specification, gimbal dependent array element look directions, focal plane detector layout, vector quantization of modulated waveform data, or the response of detector output to incident photon flux can be successful.

Governing quality metrics

See Also

PD.TileMap.WF_Returns(ia,j), SC.Array.PolCode(ia), SC.Array.RayDirs.*Ia*(icase, ie, j), SC.Array.RayDirsSigmas(ia,icase,j), SC.WF.RayDesc.ArryGlbl(iray), SC.Array.LatticeDesc(ia,j), SC.Swath.RateFactor(ia), SC.WF.Rx.Nsamps(ia), SC.WF.TxProfile.FM_Slope(ia), SC.WF.TxProfile.FM_Start(ia), SC.WF.TxProfile.FM_PhaseStart(ia), SC.WF.TxProfile.Dat(ia,j), SC.WF.TxProfile.Flags(j), SC.WF.TxProfile.Peak(ia), SC.WF.TxProfile.Nsamps(ia), SC.WF.VQ_Decode(ia,icode), SC.Array.FilterDat(ifc,j), SC.Array.PolDat(ipc,j), SC.Array.RayDirs.*Ia*(icase,ie,j) SC.Array.CurrentResp.Dat(ia,j)

20.4.10 PD.HiIndx.Artifacts [L1,2,3,4,5]

Brief Definition

PD.HiIndx.Artifacts gives the ending index values of SC.Artifacts.* and related items.

indices

none

Units/representation

Dimensionless, Int8

Treatment/usage

PD.HiIndx.Artifacts is the number of artifacts in the artifact list if PD.LoIndx.Artifacts=1.

Default when not populated:
If PD.HiIndx.Artifacts is not populated, a list of known artifacts cannot be

referenced by using applications.

Governing quality metrics

See Also SC.Artifacts.List(j)

20.4.11 PD.LoIndx.Artifacts [L1,2,3,4,5]

Brief Definition PD.LoIndx.Artifacts gives the starting index values of SC.Artifacts.* and related items.

indices none

Units/representation Dimensionless, Int8

Treatment/usage PD.LoIndx.Artifacts is the starting index (usually 0 or 1, depending on the software developer's choice) used in writing the artifacts list to the data set. PD.HiIndx.Artifacts is the number of artifacts in the artifact list if PD.LoIndx.Artifacts=1.

Default when not populated:
If PD.LoIndx.Artifacts is not populated, a list of known artifacts cannot be referenced by using applications.

Governing quality metrics

See Also

20.4.12 PD.HiIndx.Channlzn.ChRg(m) [L1,2,3,4,5]

Brief Definition PD.HiIndx.Channlzn.ChRg(m) is the highest value of channel index for channelization, m. This supports the channel index ranges in channelization history data items such as PF.Channlzn.Hist.ChTrace.*m*(j,k) with maximum range values.

indices Index, m, ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is from the lowest channelization in the processing chain[168] to the current channelization.

[168] Channel combination performed at different processing steps creates a new channelization (usually with fewer channels). The "lowest channelization" in the processing chain refers to the original collection channelization. The first output processed data set may also use the collection channelization, or it may skip to a higher number of channelization index due to the need to document significant interim and transient steps in processing that did not have separate output data sets introduced to the PED.
The "current channelization" is the one in the current data set and is the result of the channel combination history leading up to the current data set. The channelization reflects the combination state, not the number of channels.

Units/representation	Dimensionless, UInt8
Treatment/usage	PD.HiIndx.Channlzn.ChRg(m) gives the upper limit to the range of channels in channelization, m. When PD.LoIndx.Channlzn.ChRg(m) is set to 1, PD.HiIndx.Channlzn.ChRg(m) is the number of channels in channelization,m. <u>Default when not populated:</u> If PD.HiIndx.Channlzn.ChRg(m) multiple channels cannot be supported and PD.HiIndx.Channlzn.ChRg(m) should be assumed to have the same value as PD.LoIndx.Channlzn.ChRg(m).
Governing quality metrics	
See Also	PD.TileMap.ChanList(j_tile,k) PF.Channlzn.Hist.ChTrace(j,k)

20.4.13 PD.LoIndx.Channlzn.ChRg(m) [L1,2,3,4,5]

Definition	PD.LoIndx.Channlzn.ChRg(m) is the starting index for channels in channelization m. It supports the channel index ranges in channelization history data items such as PF.Channlzn.Hist.ChTrace.*m*(j,k) with minimum range values.
indices	Index, m, ranges from PD.LoIndx.Channlzn.Rg_m to PD.HiIndx.Channlzn.Rg_m, that is, from the lowest channelization in the processing chain to the current channelization.
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.LoIndx.Channlzn.ChRg(m) gives the lower limit to the range of channels in channelization, m. When PD.LoIndx.Channlzn.ChRg(m) is set to 1, PD.HiIndx.Channlzn.ChRg(m) is the number of channels in channelization,m. <u>Default when not populated:</u> If PD.LoIndx.Channlzn.ChRg(m) multiple channels cannot be supported and arrays with a range supported by PD.LoIndx.Channlzn.ChRg(m) should not be referenced.
Governing quality metrics	
See Also	

20.4.14 PD.HiIndx.Channlzn.Rg_m [L1,2,3,4,5]

Brief Definition	PD.HiIndx.Channlzn.Rg_m gives the maximum value of index, m, that

---

Each channelization has 1 or more channels. Per the discussion in PD.HiIndx.Channlzn.Rg_m, the sensor channelization in the raw data is recommended to have the value, 0.

	appears in PD.HiIndx.Channlzn.ChRg(m) and PD.LoIndx.Channlzn.ChRg(m). This is the channelization index corresponding to the current data set.
indices	none
Units/representation	Dimensionless, UInt8

Treatment/usage	PD.HiIndx.Channlzn.Rg_m, taken with PD.LoIndx.Channlzn.Rg_m indicates the number of channelizations in the processing history of the current data.
	In order to provide a one-stop source of the channelization history in the metadata for each data set in a channelized processing chain including raw sensor data, we need to account for the possibility that successive stages of processing have different numbers of channels. Generally, when there is a change from a previous channelization, the number of channels is reduced. To avoid creating arrays with many empty slots, and to make it convenient to select the channelizations and the channels within a channelization, we structure the channelization metadata history data as follows:
	• The channelizations are counted as
	  ○ channelization 0 for the channelization of the raw collected data found in an enterprise level-1 data set,
	  ○ channelization 1 for the channelization of the first post level-1 processed version of the data, {which depending on the processing may or may not be the original collected channels, and may or may not[169] be saved to a new data set}
	  ○ channelization 2 for the channelization of the second post level-1 processed version of the data, {which depending on the processing may or may not be the original collected channels, and may or may not be saved to a new data set}
	  ○ etc
	• PD.HiIndx.Channlzn.Rg_m is the value of the channelization count for the current data set and will be equal to 0 if the current data set is at enterprise level-1 and greater if the channel data has been modified in subsequent processing. For a level-2 data set that has not combined the original collection channels, PD.HiIndx.Channlzn.Rg_m would remain at 0.
	• PD.LoIndx.Channlzn.Rg_m is the value of the channelization count for the lowest channelization in the documented precursor data sets

[169] In this case, the data was in a transient interim state. It may be of significance to understanding what has been done to the data, but the processing chain is not constrained to saving it as a stand alone data set if that would have reduced system efficiency.

and will be equal to 0 if there a enterprise level-1 and greater if the data had been processed before delivery to CMMD compliant part of the processing chain .

- The value of evaluated index *m*, and index m as used in the channelization history items corresponds to a specific channelization count.
- PD.HiIndx.Channlzn.ChRg(m) and PD.LoIndx.Channlzn.ChRg(m) give the range of channels within each channelization, m.
- Each PF.Channlzn.*.*m*(j) has the range of j (over channels) given in PD.HiIndx.Channlzn.ChRg(m) and PD.LoIndx.Channlzn.ChRg(m).

Default when not populated:
If PD.HiIndx.Channlzn.Rg_m is not populated, channelization history for the current datset is not provided and it should be assumed that a single, unaltered channel of data exists throughout the processing thread for this data set. Thus, PD.HiIndx.Channlzn.Rg_m will be treated as if equal to PD.LoIndx.Channlzn.Rg_m in all default situations.

Governing quality metrics

See Also: PD.TileMap.ChanList(j_tile,k) PF.Channlzn.Hist.ChTrace(j,k)

20.4.15 PD.LoIndx.Channlzn.Rg_m [L1,2,3,4,5]

Definition: PD.LoIndx.Channlzn.Rg_m gives the minimum value of index, m, that appears in PD.HiIndx.Channlzn.ChRg(m) and PD.LoIndx.Channlzn.ChRg(m). This is the index associated with the original channelization presented to the PED.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.LoIndx.Channlzn.Rg_m is the value of the channelization index for the first channelsizstion made available to the PED and marks the starting index value for the channelization history. It is recommended that this be set to 0 when there is an enterprise level-1 data set. In a PED that has not yet employed enterprise level-1 data, the first saved data set will be for a value of PD.LoIndx.Channlzn.Rg_m =1.
See PD.HiIndx.Channlzn.Rg_m for a broader view of how the channelization history is carried in the one-stop shop CMMD compliant metadata.

Default when not populated:
If PD.LoIndx.Channlzn.Rg_m is not populated, it should be taken to be 0 or 1, whichever the implementation of the writing application prefers to indicate the minimum index for an array.

Governing quality metrics

See Also	PD.HiIndx.Channlzn.Rg_m

20.4.16 PD.HiIndx.DAL [L1,2,3,4,5]

Brief Definition	PD.HiIndx.DAL is the ending index for the list associated with the DAL versions suitable for the current data set.
indices	none
Units/representation	Dimensionless UInt8
Treatment/usage	MD.Meta.DAL_VersionName(j) is a list of DAL versions that can be used to read the current data set. PD.HiIndx.DAL is the highest value of the j for that data vector. When PD.LoIndx.DAL is set to 1, PD.HiIndx.DAL will be the number of DAL versions capable of reading the current file that are documented in the current file. Default when not populated: If PD.HiIndx.DAL is not populated, it is to be assumed that PD.HiIndx.DAL equals PD.LoIndx.DAL, it that parameter is populated. If neither item is populated, no DAL version alternates are offered and MD.Meta.DAL_VersionName(j) should not be referenced by an application reading the current data set.
Governing quality metrics	
See Also	MD.Meta.DAL_VersionName(j)

20.4.17 PD.LoIndx.DAL [L1,2,3,4,5]

Definition	PD.LoIndx.DAL is the starting index for the list associated with the DAL version listdata sets of the current data set.
indices	none
Units/representation	Dimensionless UInt8
Treatment/usage	When PD.LoIndx.DAL is set to 1, PD.HiIndx.DAL will be the number of DAL versions capable of reading the current file that are documented in the current file. Default when not populated: If PD.LoIndx.DAL is not populated, it is to be assumed that no DAL version alternates are offered and MD.Meta.DAL_VersionName(j) should not be referenced by an application reading the current data set.
Governing quality metrics	
See Also	MD.Meta.DAL_VersionName(j)

20.4.18 PD.HiIndx.Fellows[L1,2,3,4,5]

Brief Definition	PD.HiIndx.Fellows is the ending index for lists associated with the fellow data sets of the current data set.
indices	none
Units/representation	Dimensionless UInt8
Treatment/usage	When PD.LoIndx.Fellows is set to 1, PD.HiIndx.Fellows will be the number of fellow data sets documented in the current file. Only enterprise L-1, L-2, and L-3 fellows may be documented in the aggregate fellow metadata. Default when not populated: If PD.HiIndx.Fellows is not populated, PD.HiIndx.Fellows is assumed to have the value of PD.LoIndx.Fellows, if that parameter is populated. If neither is populated, a "Fellows list" is not supported and MD.Aggregate.Fellow.* should not be referenced by an application that is using the current data set.
Governing quality metrics	
See Also	MD.Aggregate.Fellow.DS_Name(j)

20.4.19 PD.LoIndx.Fellows[L1,2,3,4,5]

Definition	PD.LoIndx.Fellows is the starting index for lists associated with the fellow data sets of the current data set.
indices	none
Units/representation	Dimensionless UInt8
Treatment/usage	The choice of starting index for fellow will usually be either 1 or 0, depending on the language that the CMMD compliant data set writer is written in and the developers' taste. When PD.LoIndx.Fellows is set to 1, PD.HiIndx.Fellows will be the number of fellow data sets documented in the current file. Only enterprise L-1,L-2, and L-3 fellows may be documented in the aggregate fellow metadata. Default when not populated: If PD.LoIndx.Fellows is not populated, a "Fellows list" is not supported and MD.Aggregate.Fellow.* should not be referenced by an application that is using the current data set.
Governing quality metrics	
See Also	MD.Aggregate.Fellow.DS_Name(j)

20.4.20 PD.HiIndx.IPP(j) [L1,2,3,4,5]

Brief Definition	+
indices	Index, j, ranges from PD.LoIndx.Swath(11) to PD.HiIndx.Swath(11), that is

over the range of values for *PulseSeq*. Thus, the number of j values is the number of pulse duration sequences, which in turn are defined as repeatable IPP sequences with variable IPP duration.

Units/representation: UInt8, dimensionless

Treatment/usage: If PD.LoIndx.IPP(j)=0, then PD.HiIndx.IPP(j) is one less than the number of IPPs in the jth duration sequence defined for the data set.
We provide a Lo/HiIndx pair for each value of *PulseSeq* used in the data set. Each value of j will correspond to a conversion to an integer of a value of *PulseSeq* that appears in a tag, SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn).

Default when not populated:
If PD.HiIndx.IPP(j) is not populated, no internal pulse duration sequences are in use for the data set and SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) should not be referenced by an application using the current data set.

Governing quality metrics:

See Also: SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) PD.HiIndx.Swath(11)

20.4.21 PD.LoIndx.IPP(j) [L1,2,3,4,5]

Brief Definition: PD.LoIndx.IPP(j) supports the array index "ipn" ranges for SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) with minimum range values.

indices: Index, j, ranges from PD.LoIndx.Swath(11) to PD.HiIndx.Swath(11), that is over the range of values for *PulseSeq*. This reveals the number of pulse sequences defined as repeatable IPP sequences with variable IPP duration.

Units/representation: UInt8, dimensionless

Treatment/usage: By IPP counting convention, the index of the first IPP in a duration sequence is PD.LoIndx.IPP(j)=0. This is also the default value. We define this quantity for consistency with the practice of providing upper and lower limits for data arrays. However, population can rely on the universal default.
Since PD.LoIndx.IPP(j)=0, PD.HiIndx.IPP(j) is one less than the number of IPPs in the jth duration sequence defined for the data set.
We provide a Lo/HiIndx pair for each value of *PulseSeq* used in the data set. Each value of j will correspond to a conversion to an integer of a value of *PulseSeq* that appears in a tag, SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn).

Default when not populated:
If PD.LoIndx.IPP(j) is not populated, no internal pulse duration sequences are in use for the data set and SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) should not be referenced by an application using the current data set.

Governing quality metrics:

See Also: SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) PD.HiIndx.Swath(11)

20.4.22 PD.HiIndx.Laser(j) [L1,2,3,4,5]

Brief Definition	PD.HiIndx.Laser(j) supports the array index ranges for SC.Laser.* with maximum range values.
indices	j=1 to 3
Units/representation	UInt8, dimensionless
Treatment/usage	PD.HiIndx.Laser(1) supplies the max in the range for the index covering illuminator lasers[170] in the lidar instrument. PD.HiIndx.Laser(1) governs the first index in SC.WF.TxProfile.Dat(il,j), SC.WF.TxProfile.Peak(il), SC.WF.TxProfile.RF_Start(il), SC.WF.TxProfile.RF_PhaseStart(il), SC.WF.TxProfile.Nsamps(il), SC.WF.TxProfile.RF_Slope(il). SC.Laser.Dat(il,j), and SC.Laser.IDat(il,j). When PD.LoIndx.Laser(1) is set to 1, PD.HiIndx.Laser(1) is the number of laser illuminators in the lidar instrument that are used in distinct channels. If the lidar design employs more than one laser diffuser or transmitting 'optics chain'/'optics chain setting' and hence there is more than one beam pattern for each laser, the metadata must be populated as if each laser/Tx optics parameter set were in fact a separate laser with its specified far field beam pattern. That would require counting each (laser,laser optics chain) as a value of "il" in PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1) PD.HiIndx.Laser(2) supplies the max in the range for the index covering properties of the illuminator lasers in the lidar instrument. Governs the second index in SC.Laser.Dat(il,j). Currently should be set to 14 if PD.LoIndx.Laser(2)=1. PD.HiIndx.Laser(3) supplies the max in the range for the index covering coded properties of the illuminator lasers in the lidar instrument. Governs the second index in SC.Laser.IDat(il,j). Currently should be set to 7 if PD.LoIndx.Laser(3)=1. <u>Default when not populated:</u> If PD.HiIndx.Laser(j) is not populated, no information about the laser(s) in the lidar is populated and SC.WF.TxProfile.Dat(il,j), SC.WF.TxProfile.Peak(il), SC.WF.TxProfile.RF_Start(il), SC.WF.TxProfile.RF_PhaseStart(il), SC.WF.TxProfile.Nsamps(il), SC.WF.TxProfile.RF_Slope(il). SC.Laser.Dat(il,j), and SC.Laser.IDat(il,j) should not be referenced by the application using the current data set.

[170] If a design gangs several lasers to create the illuminator for a single channel, a single value of the "il" index applies to the illuminator and the maximum value represented here need not be raised to count individual element lasers in such an illuminator.

Governing quality metrics

See Also

SC.Laser.Dat(il,j) SC.Laser.IDat(il,j)
SC.WF.TxProfile.Dat(il,j), SC.WF.TxProfile.Peak(il),
SC.WF.TxProfile.RF_Start(il), SC.WF.TxProfile.RF_PhaseStart(il),
SC.WF.TxProfile.Nsamps(il), SC.WF.TxProfile.RF_Slope(il)

20.4.23 PD.LoIndx.Laser(j) [L1,2,3,4,5]

Brief Definition

PD.LoIndx.Laser(j) supports the array index ranges for SC.Laser.* with minimum range values.

indices

j=1 to 3

Units/representation

UInt8, dimensionless

Treatment/usage

PD.LoIndx.Laser(1)= min in the range for index covering illuminator lasers in the lidar instrument. Governs first index in SC.WF.TxProfile.Dat(il,j), SC.WF.TxProfile.Peak(il), SC.WF.TxProfile.RF_Start(il), SC.WF.TxProfile.RF_PhaseStart(il), SC.WF.TxProfile.Nsamps(il),and SC.WF.TxProfile.RF_Slope(il). Generally set to 0 or 1 per the indexing scheme and implementation language of the application writing the lidar data set.

PD.LoIndx.Laser(2) supplies the min in the range for the index covering properties of the illuminator lasers in the lidar instrument. Governs the second index in SC.Laser.Dat(il,j). Generally set to 0 or 1 per the indexing scheme and implementation language of the application writing the lidar data set.

PD.LoIndx.Laser(3) supplies the min in the range for the index covering coded properties of the illuminator lasers in the lidar instrument. Governs the second index in SC.Laser.IDat(il,j). Generally set to 0 or 1 per the indexing scheme and implementation language of the application writing the lidar data set.

The 0:N-1 vs 1:N decision is made in L-1 writer. May override L-0 numbering as long as consistent data tracking is assured.

Default when not populated:

If PD.LoIndx.Laser(j) is not populated, no information about the laser(s) in the lidar is populated and SC.WF.TxProfile.Dat(il,j), SC.WF.TxProfile.Peak(il), SC.WF.TxProfile.RF_Start(il), SC.WF.TxProfile.RF_PhaseStart(il), SC.WF.TxProfile.Nsamps(il), SC.WF.TxProfile.RF_Slope(il). SC.Laser.Dat(il,j), and SC.Laser.IDat(il,j) should not be referenced by the application using the current data set.

Governing quality metrics

See Also

20.4.24 PD.HiIndx.MD_Dat [L1,2,3,4,5]

Brief Definition	PD.HiIndx.MD_Dat gives the end of the "j" index range for MD.UseHist.* data items
indices	none
Units/representation	UInt8, dimensionless
Treatment/usage	When PD.LoIndx.MD_Dat is set to 1, PD.HiIndx.MD_Dat is the number of elements in the PD.UseHist.*(j) data vectors, that is the number of reported access events. <u>Default when not populated:</u> The default value of PD.HiIndx.MD_Dat is the value of PD.LoIndx.MD_Dat, which could itself be defaulted. If PD.HiIndx.MD_Dat is not populated, either MD.UseHist.*(j) should not be populated, or it can document only a single access event as the range of j is assumed to have only one value of j as a default.
Governing quality metrics	
See Also	

20.4.25 PD.LoIndx.MD_Dat [L1,2,3,4,5]

Brief Definition	PD.LoIndx.MD_Dat gives the start of the "j" index range for MD.UseHist.* data items
indices	none
Units/representation	UInt8, dimensionless
Treatment/usage	When PD.LoIndx.MD_Dat is set to 1, PD.HiIndx.MD_Dat is the number of elements in the PD.UseHist.*(j) data vectors, that is the number of reported access events. <u>Default when not populated:</u> If PD.LoIndx.MD_Dat is not populated, it will be set to 0 or 1 depending on the coding language or convention of the writing application creating the current dataset.
Governing quality metrics	
See Also	

20.4.26 PD.HiIndx.MD_Strings [L1,2,3,4,5]

Brief Definition	PD.HiIndx.MD_Strings gives end of the "j" index range for PD.Wordsize.Meta(j)
indices	none
Units/representation	UInt8, dimensionless
Treatment/usage	When PD.HiIndx.MD_Strings is set to 1, PD.HiIndx.MD_Strings is the

number of elements in the PD.Wordsize.Meta(j) data vector, which is used to give the number of characters in various MD.* strings.
If PD.LoIndx.MD_Strings=1, currently PD.HiIndx.MD_Strings should be set to 3.

Default when not populated:
If PD.HiIndx.MD_Strings is not populated, an error condition should be flagged. PD.HiIndx.MD_Strings could be set to the value indicated as currently correct in the above text. However, this value may change in future versions of the CMMD. If a data set is populated with more "sized" metadata strings than the older default value used, then the previous default value determines the number of wordsizes provided. Added (newer) metadata items would be ignored. If a data set is populated with fewer than the back-stop default value above {the case when processing an old data set with a code written per a future 'then current' CMMD provided default value for PD.HiIndx.MD_Strings}, then the application ingesting the current data set might find fewer populated metadata strings than expected and blank fill the slots for unsupported strings. The error condition flag can inform users and operators of the situation and the path taken. To be current, population of PD.HiIndx.MD_Strings should not rely on any defaults and explicit population should be implemented.

Governing quality metrics

See Also PD.Wordsize.Meta(j)

20.4.27 PD.LoIndx.MD_Strings [L1,2,3,4,5]

Brief Definition PD.LoIndx.MD_Strings gives start of the index range for PD.Wordsize.Meta(j)

indices none

Units/representation UInt8, dimensionless

Treatment/usage When PD.LoIndx.MD_Strings is set to 1, PD.HiIndx.MD_Strings is the number of elements in the PD.Wordsize.Meta(j) data vector, which is used to give the number of characters in various MD.* strings.

Default when not populated:
If PD.LoIndx.MD_Strings is not populated, it should be assumed to have the value of 1 so that PD.LoIndx.MD_Strings, when populated, will be the number of strings whose wordsizes are provided.

Governing quality metrics

See Also PD.Wordsize.Meta(j)

20.4.28 PD.HiIndx.Overlap(j) [L1,2]

Brief Definition	PD.HiIndx.Overlap(j) supports the array index ranges for SC.Overlap.* with maximum range values.
indices	j=1 to 4
Units/representation	UInt32, dimensionless
Treatment/usage	j=1 max in the range for the index covering pulses in a scan for overlap parameters. PD.HiIndx.Overlap(1) governs first index in SC.Overlap.Pulse.X_Hi(j,k), SC.Overlap.Pulse.X_Lo(j,k), SC.Overlap.Pulse.Y_Hi(j,k), and SC.Overlap.Pulse.Y_Lo(j,k) j=2 max in the range for the index covering scans in a subswath for overlap parameters. PD.HiIndx.Overlap(2) governs second index in SC.Overlap.Pulse.X_Hi(j,k), SC.Overlap.Pulse.X_Lo(j,k), SC.Overlap.Pulse.Y_Hi(j,k), and SC.Overlap.Pulse.Y_Lo(j,k) j=3 max in the range for index covering scan pairs as intended to be collected in a list for registration. PD.HiIndx.Overlap(3) governs the index for SC.Overlap.Scan.ID1(j), SC.Overlap.Scan.ID2(j), SC.Overlap.Scan.X_Hi(j), SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.Y_Hi(j), SC.Overlap.Scan.Y_Lo(j), and first index of PF.Adjustmts.Mean.Scan(is, j). j=4 max in the range of the index covering subswath pairs as processed. PD.HiIndx.Overlap(4) governs the index for PF.Overlap.SubSwath.ID1(j), PF.Overlap.SubSwath.ID2(j), PF.Overlap.SubSwath.IDL1(j), PF.Overlap.SubSwath.IDL2(j), PF.Adjustmts.Mean.Subswath(iswp, j) <u>Default when not populated:</u> If the PD.HiIndx.Overlap(j) array is not populated, no overlap data is provided and no registration dependent on knowing the nominal overlap of its input patches should be executed. When any element of the array is not known but at least one element is being populated, the unknown value should be set to 4294967295, the maximum value of a 32 bit unsigned integer.
Governing quality metrics	
See Also	PD.LoIndx.Overlap(j) PF.Adjustmts.Mean.Scan(is, j). SC.Overlap.Pulse.X_Hi(j,k), SC.Overlap.Pulse.X_Lo(j,k), SC.Overlap.Scan.ID1(j), SC.Overlap.Scan.X_Hi(j), SC.Overlap.Scan.X_Lo(j), PF.Overlap.SubSwath.ID1(j) PF.Adjustmts.Mean.Subswath(isw, j)

20.4.29 PD.LoIndx.Overlap(j) [L1,2]

Definition: PD.LoIndx.Overlap(j) supports the array index ranges for SC.Overlap.* and PF.Overlap.* with minimum range values.

indices: j=1,4

Units/representation: UInt8, dimensionless

Treatment/usage: j=1 min in the range for index covering pulses in a scan for overlap parameters. Governs first index in SC.Overlap.Pulse.X_Hi(j,k), SC.Overlap.Pulse.X_Lo(j,k), SC.Overlap.Pulse.Y_Hi(j,k), and SC.Overlap.Pulse.Y_Lo(j,k)

j=2 min in the range for index covering scans in a subswath for overlap parameters. Governs second index in SC.Overlap.Pulse.X_Hi(j,k), SC.Overlap.Pulse.X_Lo(j,k), SC.Overlap.Pulse.Y_Hi(j,k), and SC.Overlap.Pulse.Y_Lo(j,k)

j=3 min in the range for the index covering scan pairs as intended to be collected in a list for registration. Governs index for SC.Overlap.Scan.ID1(j), SC.Overlap.Scan.ID2(j), SC.Overlap.Scan.X_Hi(j), SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.Y_Hi(j), and SC.Overlap.Scan.Y_Lo(j).

j=4 min in the range of the index covering subswath pairs as processed. Governs the index for PF.Overlap.SubSwath.ID1(j), PF.Overlap.SubSwath.ID2(j), PF.Overlap.SubSwath.IDL1(j), PF.Overlap.SubSwath.IDL2(j), PF.Adjustmts.Mean.Subswath(iswp, j)

<u>Default when not populated:</u>
If the PD.LoIndx.Overlap(j) array is not populated, no overlap data is provided and no registration dependent on knowing the nominal overlap of its input patches should be executed. When any element of the array is not known but at least one element is being populated, the unknown values should be set to 1 to start the array such that PD.HiIndx.Overlap(j) provides the number of items [pulses, scans, subswaths] (to be) registered.

Governing quality metrics

See Also

20.4.30 PD.HiIndx.Parents[L1,2,3,4,5]

Brief Definition: PD.HiIndx.Parents is the ending index for lists associated with the parent data

sets of the current data set.

indices: none

Units/representation: Dimensionless UInt8

Treatment/usage: All enterprise level-2 and level-3 data sets have a single parent data set. Enterprise level-4 data sets have 2 or more parent data sets from which their wide band content is derived. If PD.LoIndx.Parents is set to 1, PD.HiIndx.Parents is the number of parent data sets for a given data set. For level-2 and level-3 data sets, PD.HiIndx.Parents= PD.LoIndx.Parents, and for level-4 data sets PD.HiIndx.Parents> PD.LoIndx.Parents. This data item is used to govern PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) and MD.Aggregate.Parent.*.
While level-1 data sets typically have several sources, they are all level-0 and will never be operationally accessed or described in a PED architecture. We would like to be able to identify information about the level-1 writer for data sets at any level and we can do this with the same data tag (without a special version for level-1 or level-4). The strategy requires that we be able to supply PD.HiIndx.Parents at all levels. Since we use the same MD.ProcHist.* items at level-1 as at level 2, 3 and 4, we need to have an index range of 1 defined for k at level-1 because L-1 data sets are the single root PED data sets for single collect data sets. This is supplied by PD.Lo/HiIndx.Parents for level-1 just as it is for L-2 and L-3. Only L-4 data sets get a larger range.

Default when not populated:
If PD.HiIndx.Parents or PD.LoIndx.Parents are not populated at enterprise levels 2 or 3, then they can be assumed to be equal to each other with a value of 1.

Governing quality metrics:

See Also: PD.TileMap.StartPulseNum.*TileNum*(j_parent)
MD.Aggregate.Parent.DS_Name(j)

20.4.31 PD.LoIndx.Parents [L1,2,3,4,5]

Definition: PD.LoIndx.Parents is the starting index for lists associated with the parent data sets of the current data set.

indices: none

Units/representation: Dimensionless UInt8

Treatment/usage: All enterprise level-2 and level-3 data sets have a single parent data set. Enterprise level-4 data sets have 2 or more parent data sets from which their wide band content is derived. If PD.LoIndx.Parents is set to 1, PD.HiIndx.Parents is the number of parent data sets for a given data set. This data item is used to govern PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)

and MD.Aggregate.Parent.* as well as MD.ProcHist.*.
PD.LoIndx.Parents is part of a data design that permits the same metadata item to be defined to supply information about the level-1 writer application for enterprise level-1 (and 2,3) and level-4 data sets. See the discussion in the paragraph above for PD.HiIndx.Parents.

Default when not populated:
If PD.HiIndx.Parents or PD.LoIndx.Parents are not populated at enterprise levels 2, then they can be assumed to be equal to each other with a value of 1.

Governing quality metrics

See Also

PD.Pulse.StartPulseNum.*TileNum*(j_parent)
MD.Aggregate.Parent.DS_Name(j) PD.HiIndx.Parents

20.4.32 PD.HiIndx.Point.RangeListC.*OrdIndx* [L2,3,4,5]

Definition

PD.HiIndx.Point.RangeListC.*OrdIndx*[171] provides the ending value for the first index of PD.Point.RangeListC.*OrdIndx*(j,k).

indices

None.
OrdIndx would be a string that looks like an integer number that maps to a particular row value in ES.Point.SigmaXYZ(j,k). As such, *OrdIndx* acts much like an index, even though it is part of the parameter name.

Units/representation

Dimensionless, UInt32

Treatment/usage

We consider the likely case that many points in a lidar data set have approximately the same earth frame error covariance matrix. Such covariances can be represented by a single common value of the error covariance matrix. These points will generally lie in extended sequences within the data set tiles. Since the number of common covariance matrix values and the number of point ranges with a single common covariance matrix is a variable, we need to be able to list all of the point ranges in any data set associated with a common value of the error covariance matrix. PD.HiIndx.Point.RangeListC.*OrdIndx* is the ending value of index, j, in the list, PD.Point.RangeListC.*OrdIndx*(j,k). When PD.LoIndx.Point.RangeListC.*OrdIndx* is set to 1, PD.HiIndx.Point.RangeListC.*OrdIndx* will be the number of point ranges that have the common covariance matrix value indicated by *OrdIndx*, the ordinal index for the individual values of the lidar point error covariance matrix.

[171] The seemingly superfluous, “point” is made part of this tag to support the possiblilty of wild card requests to the DAL. This item would likely be part a group of metadata items used together.

Default when not populated:
If PD.HiIndx.Point.RangeListC.*OrdIndx* is not populated, no pre-computed error covariance is provided for points. PD.Point.RangeListC.*OrdIndx*(j,k) should not be referenced by any application using the current data set.

Governing quality metrics: Must be populated with the correct number.

See Also: PD.Point.RangeListC.*OrdIndx*(j,k) ES.Point.SigmaXYZ(j,k)

20.4.33 PD.LoIndx.Point.RangeListC.*OrdIndx* [L2,3,4,5]

Definition: PD.LoIndx.Point.RangeListC.*OrdIndx* provides the starting value for the first index of PD.Point.RangeListC.*OrdIndx*(j,k).

indices: None.

OrdIndx would be a string that looks like an integer number that maps to a particular row value in ES.Point.SigmaXYZ(j,k). As such, *OrdIndx* acts much like an index, even though it is part of the parameter name.

Units/representation: Dimensionless, UInt8

Treatment/usage: We consider the likely case that many points in a lidar data set have approximately the same earth frame error covariance matrix. Such covariances can be represented by a single common value of the error covariance matrix. These points will generally lie in extended sequences within the data set tiles. Since the number of common covariance matrix values and the number of point ranges with a single common covariance matrix is a variable, we need to be able to list all of the point ranges in any data set associated with a common value of the error covariance matrix. PD.LoIndx.Point.RangeListC.*OrdIndx* is the starting value of index, j, in the list, PD.Point.RangeListC.*OrdIndx*(j,k).
PD.LoIndx.Point.RangeListC.*OrdIndx* is typically set to 0 or 1.

Default when not populated:
If PD.LoIndx.Point.RangeListC.*OrdIndx* is not populated, no pre-computed error covariance is provided for points. PD.Point.RangeListC.*OrdIndx*(j,k) should not be referenced by any application using the current data set.

Governing quality metrics: Must be populated with a value that points to the first value.

See Also: PD.Point.RangeListC.*OrdIndx*(j,k) ES.Point.SigmaXYZ(j,k)

20.4.34 PD.HiIndx.Point.RangeListX.*OrdIndx* [L2,3,4,5]

Definition: PD.HiIndx.Point.RangeListX.*OrdIndx* provides the ending value for the first index of PD.Point.RangeListX.*OrdIndx*(j,k).

indices: None.

	OrdIndx would be a string that looks like an integer number that maps to a particular row value in ES.Point.SigmaXYZ(j,k). As such, *OrdIndx* acts much like an index, even though it is part of the parameter name.
Units/representation	Dimensionless, UInt32
Treatment/usage	We consider the likely case that many points in a lidar data set have approximately the same earth frame error covariance matrix. Such covariances can be represented by a single value of the error covariance matrix, even if they serve a minority of the points in a data set. "Common covariance" points will generally lie in extended sequences within the data set tiles and "exceptional covariance" points lie in shorter sequences embedded within the "common covariance" sequences. Since the number of exceptional covariance matrix values and the number of point ranges with a single exceptional covariance matrix is a variable, we need to be able to list all of the point ranges in any data set associated with an exceptiopnal value of the error covariance matrix. PD.HiIndx.Point.RangeListX.*OrdIndx* is the ending value of index, j, in the list, PD.Point.RangeListX.*OrdIndx*(j,k). When PD.LoIndx.Point.RangeListX.*OrdIndx* is set to 1, PD.HiIndx.Point.RangeListX.*OrdIndx* will be the number of point ranges that have the exceptional covariance matrix value indicated by *OrdIndx*, the ordinal index for the individual values of the lidar point error covariance matrix. <u>Default when not populated:</u> If PD.HiIndx.Point.RangeListX.*OrdIndx* is not populated, no pre-computed error covariance exceptions are provided for points. PD.Point.RangeListX.*OrdIndx*(j,k) should not be referenced by any application using the current data set.
Governing quality metrics	Must be populated with the correct number.
See Also	PD.Point.RangeListX.*OrdIndx*(j,k) ES.Point.SigmaXYZ(j,k)

20.4.35 PD.LoIndx.Point.RangeListX.*OrdIndx*[L2,3,4,5]

Definition	PD.LoIndx.Point.RangeListX.*OrdIndx* provides the starting value for the first index of PD.Point.RangeListX.*OrdIndx*(j,k).
indices	None.
	OrdIndx would be a string that looks like an integer number that maps to a particular row value in ES.Point.SigmaXYZ(j,k). As such, *OrdIndx* acts much like an index, even though it is part of the parameter name.
Units/representation	Dimensionless UInt8
Treatment/usage	We consider the likely case that many points in a lidar data set have approximately the same earth frame error covariance matrix. Such

covariances can be represented by a single common value of the error covariance matrix. These points will generally lie in extended sequences within the data set tiles. There may be short subsequences where an exceptional value of the covariance occurs. Since the number of exceptional covariance matrix values and the number of point ranges with a single exceptional covariance matrix is a variable, we need to be able to list all of the point ranges in any data set associated with an exceptional value of the error covariance matrix. PD.LoIndx.Point.RangeListX.*OrdIndx* is the starting value of index, j, in the list, PD.Point.RangeListX.*OrdIndx*(j,k). PD.LoIndx.Point.RangeListX.*OrdIndx* is typically set to 0 or 1.

Default when not populated:
If PD.LoIndx.Point.RangeListX.*OrdIndx* is not populated, no pre-computed error covariance exceptions are provided for points. PD.Point.RangeListX.*OrdIndx*(j,k) should not be referenced by any application using the current data set.

Governing quality metrics: Must be populated with the correct number.

See Also: PD.Point.RangeListX.*OrdIndx*(j,k) ES.Point.SigmaXYZ(j,k)

20.4.36 PD.HiIndx.Point.SigmaDat [L2,3,4,5]

Brief Definition: PD.HiIndx.Point.SigmaDat is the ending index for ES.Point.SigmaXYZ(j,k) first index

indices: none

Units/representation: Dimensionless, UInt16

Treatment/usage: Pre-computed error ellipsoids are stored by value (a 6-tuple composed of 3 values of variance and 3 values of correlation) and assigned to points by a range of "point rows" within tiles. [Note that a point row does not carry an explicit metadata parameter to act as the row number, rather it is a dummy index in a virtual array that can be thought of like a database.]
PD.HiIndx.Point.SigmaDat is the ending value for the first index of the list of error ellipsoids in ES.Point.SigmaXYZ(j,k). It also serves to quantify the maximum value that gets encoded into substrings "*OrdIndx*", which act as part of the name for the common and exceptions point interval lists.

Default when not populated:
If PD.HiIndx.Point.SigmaDat is not populated, no pre-computed errors are provided and ES.Point.SigmaXYZ(j,k) should not be referenced by any aplication using the current data set.

Governing quality metrics

See Also	ES.Point.SigmaXYZ(j,k) PD.Point.RangeListC.*OrdIndx*(j,k) PD.Point.RangeListX.*OrdIndx*(j,k)

20.4.37 PD.LoIndx.Point.SigmaDat [L2,3,4,5]

Brief Definition	PD.LoIndx.Point.SigmaDat is the starting index for ES.Point.SigmaXYZ(j,k) first index
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	Pre-computed error ellipsoids are stored by value (a 6-tuple composed of 3 values of variance and 3 values of correlation) and assigned to points by a range of "point rows" within tiles. [Note that a point row does not carry an explicit metadata parameter to act as the row number, rather it is a dummy index in a virtual array that can be thought of like a database.] PD.LoIndx.Point.SigmaDat is the starting value for the first index of the list of error ellipsoids in ES.Point.SigmaXYZ(j,k). It also serves to quantify the minimum value that gets encoded into substrings "*OrdIndx*", which act as part of the name for the common and exceptions point interval lists. <u>Default when not populated:</u> If PD.LoIndx.Point.SigmaDat is not populated, no pre-computed errors are provided and ES.Point.SigmaXYZ(j,k) should not be referenced by any aplication using the current data set.
Governing quality metrics	
See Also	ES.Point.SigmaXYZ(j,k) PD.Point.RangeListC.*OrdIndx*(j,k) PD.Point.RangeListX.*OrdIndx*(j,k)

20.4.38 PD.HiIndx.Point.Tile [L2,3,4,5]

Brief Definition	PD.HiIndx.Point.Tile gives the last value of the sequence of the j_tile index for tiles associated with point data.
indices	none
Units/representation	Dimensionless, UInt32
Treatment/usage	In the list that associates an integer index to each evaluation of *TileNum*, there may be a subsequence for point related data. PD.HiIndx.Point.Tile[172] is the highest value of j_tile in PD.TileMap.TileNum(j_tile) associated with point

[172] Note this usage is subtley different than most Lo/HiIndx pairs. This defines usage within the range of an index, not what the outer values of index are.

data bearing tiles. Used to govern, for example, PD.TileMap.X_Range(j_tile,k).

Default when not populated:
If PD.HiIndx.Point.Tile is not populated, there is no point data in tiles. Items such as PD.TileMap.X_Range(j_tile,k) whose index ranges are governed by PD.HiIndx.Point.Tile should not be referenced by an application using the current data set.

Governing quality metrics: Must be the correct value.

See Also: PD.TileMap.TileNum(j_tile)

20.4.39 PD.LoIndx.Point.Tile [L2,3,4,5]

Brief Definition: PD.LoIndx.Point.Tile gives the first value of the sequence of the j_tile index for tiles associated with point data.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: In the list that associates an integer index to each evaluation of *TileNum*, there may be a subsequence for point related data. PD.LoIndx.Point.Tile[172] is the lowest value of j_tile in PD.TileMap.TileNum(j_tile) associated with point data bearing tiles. Used to govern, for example, PD.TileMap.X_Range(j_tile,k).

Default when not populated:
If PD.LoIndx.Point.Tile is not populated, there is no point data in tiles. Items such PD.TileMap.X_Range(j_tile,k) whose index ranges are governed by PD.HiIndx.Point.Tile should not be referenced by an application using the current data set.

Governing quality metrics: Must be the correct value.

See Also: PD.TileMap.TileNum(j_tile)

20.4.40 PD.HiIndx.PrevStages(k) [L2,3,4,5]

Brief Definition: PD.HiIndx.PrevStages(k) gives the last value of the index, mm, in PF.Recon.PrevProdNames(mm,k)

indices: Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. For data sets at enterprise level-2,3 it will be the case that PD.LoIndx.Parents = PD.HiIndx.Parents because there is only one parent. For level-4 data sets, PD.HiIndx.Parents is greater than PD.LoIndx.Parents.

Units/representation: Dimensionless, UInt8

Treatment/usage: This supports documenting which of any precursor data stages actually were saved as products. PF.Recon.PrevProdNames(mm,k) gives the data set name of the mmth precursor product. Thus, if PD.LoIndx.PrevStages(k) is set to 1,

PD.HiIndx.PrevStages(k) is the number of previous products in the processing chain leading to the current data set per parent. This accounting permits that there may have been multiple precursors at enterprise levels 2, 3, or 4. In fact, this value would be the sum over applicable values of *n* of MD.ProcHist.L*n*NumStages + 1[if there was an enterprise level-1 data set used in the processing chain], had all of these items been populated.

Default when not populated:
If PD.HiIndx.PrevStages(k) is not populated, the ability to support regeneration of the current data set from earlier precursors is not supported and PF.Recon.PrevProdNames(mm,k) should not be referenced by an application using the current data set as a data source.

Governing quality metrics

See Also: PF.Recon.PrevProdNames(mm,k) MD.ProcHist.L2NumStages

20.4.41 PD.LoIndx.PrevStages(k) [L2,3,4,5]

Brief Definition: PD.LoIndx.PrevStages(k) gives the first value of the index in PF.Recon.PrevProdNames(mm,k)

indices: Index, k, ranges from PD.LoIndx.Parents to PD.HiIndx.Parents, that is over the number of parents for the current data set. For data sets at enterprise level-2,3 it will be the case that PD.LoIndx.Parents = PD.HiIndx.Parents because there is only one parent. For level-4 data sets, PD.HiIndx.Parents is greater than PD.LoIndx.Parents.

Units/representation: Dimensionless, UInt8

Treatment/usage: This supports documenting which of any precursor data stages actually were saved as products. PF.Recon.PrevProdNames(mm,k) gives the data set name of the mmth precursor product. Thus, if PD.LoIndx.PrevStages(k) is set to 1, PD.HiIndx.PrevStages(k) is the number of previous products in the processing chain leading to the current data set per parent.

Default when not populated:
If PD.LoIndx.PrevStages(k) is not populated, the ability to support regeneration of the current data set from earlier precursors is not supported and PF.Recon.PrevProdNames(mm,k) should not be referenced by an application using the current data set as a data source.

Governing quality metrics

See Also: PF.Recon.PrevProdNames(mm,k)

20.4.42 PD.HiIndx.Pulse [L1,2,3,4,5]

Brief Definition: PD.HiIndx.Pulse gives the last value of the sequence of the j_tile index for

	tiles associated with Pulse data.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	In the list that associates an integer index to each evaluation of *TileNum*, there is likely to be a sub-sequence for pulse related data. PD.HiIndx.Pulse[172] is the highest value of j_tile in PD.TileMap.TileNum(j_tile) associated with pulse data bearing tiles. Used to govern PD.TileMap.PulseRange(j_tile,k). <u>Default when not populated:</u> Essential at enterprise L-2.
Governing quality metrics	Must be the correct value.
See Also	PD.TileMap.TileNum(j_tile) PD.TileMap.PulseRange(j_tile,k)

20.4.43 PD.LoIndx.Pulse [L1,2,3,4,5]

Brief Definition	PD.LoIndx.Pulse gives the first value of the sequence of the j_tile index for tiles associated with Pulse data.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	In the list that associates an integer index to each evaluation of *TileNum*, there is likely to be a sub-sequence for pulse related data. PD.LoIndx.Pulse[172] is the lowest value of j_tile in PD.TileMap.TileNum(j_tile) associated with pulse data bearing tiles. Used to govern PD.TileMap.PulseRange(j_tile,k).
Governing quality metrics	Must be the correct value.
See Also	PD.TileMap.TileNum(j_tile)

20.4.44 PD.HiIndx.Quench[L1,2]

Brief Definition	PD.HiIndx.Quench provides the high value of the index for the array, SC.Quench.Times(j).
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	To support a transport independent data model, the range of the index for any variable sized array is made available as a metadata item. This enables us to have the same form of array support for quench time data as any other lidar metadata variable sized array. If PD.LoIndx.Quench is set to 1, then PD.HiIndx.Quench would be set to the number of quenches per IPP receive window. <u>Default when not populated:</u> If PD.HiIndx.Quench is not populated, a multi-quench GM sensor is not being described and SC.Quench.Times(j) should not be referenced by an application

using the current data set.

Governing quality metrics: Correct value is populated.

See Also: PD.LoIndx.Quench SC.Quench.times(j)

20.4.45 PD.LoIndx.Quench[L1,2]

Brief Definition: PD.LoIndx.Quench provides the low value of the index for the array, SC.Quench.Times(j).

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: To support a transport indepependent data model, the range of the index for any variable sized array is made available as a metadata item. This enables us to have the same form of array support for quench time data as any other lidar metadata variable sized array.

Default when not populated:
If PD.LoIndx.Quench is not populated, a multi-quench GM sensor is not being described and SC.Quench.Times(j) should not be referenced by an application using the current data set.

Governing quality metrics: Correct value is populated.

See Also: PD.HiIndx.Quench SC.Quench.times(j)

20.4.46 PD.HiIndx.Ray.*TileNum* [L1,2]

Brief Definition: PD.HiIndx.Ray.*TileNum* gives the tile dependent ending value for index, iray, which covers the look directions for a given array.

indices: None, however, *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile.

Units/representation: Dimensionless, UInt32

Treatment/usage: When SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray) and are used instead of SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray), the ablilty to match the range of iray to the needs of each tile is required. While individual Enterprise level-1 and level-2 writers will tend to populate PD.LoIndx.Ray.*TileNum* to either 0 or 1 all of the time, PD.HiIndx.Ray.*TileNum* must be set to SC.Ray.LoIndx.*TileNum* +(N-1) where N is the number of rays in the tile identified by *TileNum*.

Default when not populated:
PD.HiIndx.Ray.*TileNum* is essential at enterprise level-1. If PD.HiIndx.Ray.*TileNum* is not populated at enterprise level-2, array element

dependent data is not supported and SC.WF.RayDesc.Arry.*TileNum*(iray), SC.WF.RayDesc.Elmt.*TileNum*(iray), SC.WF.RayDesc.ArryGlbl(iray), and SC.WF.RayDesc.ElmtGlbl(iray) should not be referenced by an application using the the current data set.

Governing quality metrics: Consistent population of values with all other references to "iray" based indexing for a given data set.

See Also: SC.WF.RayDesc.Arry.*TileNum*(iray) SC.WF.RayDesc.Elmt.*TileNum*(iray)

20.4.47 PD.LoIndx.Ray.*TileNum*[L1,2]

Brief Definition: PD.LoIndx.Ray.*TileNum* gives the tile dependent starting value for index, iray, which covers the look directions for a given array.

indices: None, however, *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile.

Units/representation: Dimensionless UInt8

Treatment/usage: When SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray) and are used instead of SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray), the ablilty to match the range of iray to the needs of each tile is required. While individual Enterprise level-1 and level-2 writers will tend to populate PD.LoIndx.Ray.*TileNum* to either 0 or 1 all of the time, retaining PD.LoIndx.Ray.*TileNum* supports making the metadata support structurally consistent with most other arrays in this conceptual model.

Default when not populated:
PD.LoIndx.Ray.*TileNum* is essential at enterprise level-1. If PD.LoIndx.Ray.*TileNum* is not populated at enterprise level-2, array element dependent data is not supported and SC.WF.RayDesc.Arry.*TileNum*(iray), SC.WF.RayDesc.Elmt.*TileNum*(iray), SC.WF.RayDesc.ArryGlbl(iray), and SC.WF.RayDesc.ElmtGlbl(iray) should not be referenced by an application using the the current data set.

Governing quality metrics: Consistent population of values with all other references to "iray" based indexing for a given data set.

See Also: SC.WF.RayDesc.Arry.*TileNum*(iray) SC.WF.RayDesc.Elmt.*TileNum*(iray)

20.4.48 PD.HiIndx.Recon(j) [L1,2,3,4,5]

Brief Definition: PD.HiIndx.Recon(j) provides the ending values of the reconstruction data index ranges.

indices: j=1,2,3

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.HiIndx.Recon(1) gives the maximum value of the index that ranges over the processing steps/applications executed in the product generation chain. If

PD.LoIndx.Recon(1) is set to 1, then PD.HiIndx.Recon(1) is the number of applications that have been executed to create the product sequence culminating in the current data set. PD.HiIndx.Recon(1) governs, for example, PF.Recon.App(j).
PD.HiIndx.Recon(2) gives the maximum value of the index that ranges over the processing sites where the data in the chain leading to the current data set have been processed. Each site is permitted its own unique naming of processing algorithm applications. If PD.LoIndx.Recon(2) is set to 1, then PD.HiIndx.Recon(2) is the number of sites where applications have been executed to create the product sequence culminating in the current data set. PD.LoIndx.Recon(2) governs PF.Recon.Site(j).
PD.HiIndx.Recon(3) gives the maximum value of the index that ranges over twice the number of root[173] wide band data files used to create the current data set. For each root data set, both a data set name and a name for the archive where the root data set can be found is provided. If PD.LoIndx.Recon(3) is set to 1, then PD.HiIndx.Recon(3) is twice the number of data sets contributing to wideband data in the current data set. PD.HiIndx.Recon(3) governs PF.Recon.RootDS(j).

Default when not populated:
If PD.HiIndx.Recon(j) is not populated, the processing pedigree is not supported. Reconstruction data such as PF.Recon.App(j), PF.Recon.Site(j), and PF.Recon.RootDS(j) should not be referenced by applications using the current data set.

Governing quality metrics

See Also: PF.Recon.App(j) PF.Recon.Site(j) PF.Recon.RootDS(j)

20.4.49 PD.LoIndx.Recon(j) [L1,2,3,4,5]

Brief Definition: PD.LoIndx.Recon(j) provides the low value of reconstruction data index ranges.

indices: j=1,2,3

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.LoIndx.Recon(1) gives the minimum value of the index that ranges over the processing steps/applications executed in the product generation chain. If PD.LoIndx.Recon(1) is set to 1, then PD.HiIndx.Recon(1) is the number of

[173] A root data set is the first data set in a contributing processing path to the current product. Ideally, this would always be at enterprise level-1, but this is not necessary. A current data set at enterprise level-2 or level-3 will have a single root data set and it will be a less processed data set, even if it is at the same product level as the current data set. If the current data set is a level-4 fusion of multiple level-3 products, then there may be 2 or more contributing processing paths and thus 2 or more root data sets for the current data set.

applications that have been executed to create the product sequence culminating in the current data set. Governs, for example, PF.Recon.App(j). PD.LoIndx.Recon(2) gives the minimum value of the index that ranges over the processing sites where the data in the chain leading to the current data set have been processed. If PD.LoIndx.Recon(2) is set to 1, then PD.HiIndx.Recon(2) is the number of sites where applications have been executed to create the product sequence culminating in the current data set. PD.HiIndx.Recon(3) gives the minimum value of the index that ranges over twice the number of root wide band data files used to create the current data set. For each root data set, both a data set name and a name for the archive where the root data set can be found is provided. If PD.LoIndx.Recon(3) is set to 1, then PD.HiIndx.Recon(3) is twice the number of data sets contributing to wideband data in the current data set.

Default when not populated:
If PD.LoIndx.Recon(j) is not populated, the processing pedigree is not supported. Reconstruction data such as PF.Recon.App(j), PF.Recon.Site(j), and PF.Recon.RootDS(j) should not be referenced by applications using the current data set.

Governing quality metrics

See Also: PF.Recon.App(j) PF.Recon.Site(j) PF.Recon.RootDS(j) PD.HiIndx.Recon(j)

20.4.50 PD.HiIndx.Remap [L2,3,4,5]

Brief Definition: PD.HiIndx.Remap provides the high value of the first index of ES.Point.IntnsyRemap(j,k)

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: ES.Point.IntnsyRemap(j,k) lists a pair of values (k=1,2) for each segment of a piecewise linear remapping function. PD.HiIndx.Remap indicates the ending value of the index, j, in ES.Point.IntnsyRemap(j,k). When PD.LoIndx.Remap is set to 1, PD.HiIndx.Remap is the number of linear segments in the intensity remapping function.

Default when not populated:
If PD.HiIndx.Remap is not populated, an intensity remap is not supported for points and ES.Point.IntnsyRemap(j,k) should not be referenced by an application that processes the current data set.

Governing quality metrics

See Also: ES.Point.IntnsyRemap(j,k)

20.4.51 PD.LoIndx.Remap [L2,3,4,5]

Brief Definition	PD.LoIndx.Remap provides the low value of the first index of ES.Point.IntnsyRemap(j,k)
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	ES.Point.IntnsyRemap(j,k) lists a pair of values (k=1,2) for each segment of a piecewise linear remapping function. PD.HiIndx.Remap indicates the ending value of the index, j, in ES.Point.IntnsyRemap(j,k). When PD.LoIndx.Remap is set to 1, PD.HiIndx.Remap is the number of linear segments in the intensity remapping function. <u>Default when not populated:</u> If PD.LoIndx.Remap is not populated, an intensity remap is not supported for points and ES.Point.IntnsyRemap(j,k) should not be referenced by an application that processes the current data set.
Governing quality metrics	
See Also	ES.Point.IntnsyRemap(j,k)

20.4.52 PD.HiIndx.RgAmb(j) [L1,2,3,4,5]

Brief Definition	PD.HiIndx.RgAmb(j) determines the high value of the indices for the array, SC.RgAmb.Num.*Il*(isnc,j).
indices	Index,j=1,2
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.HiIndx.RgAmb(1)= end value for indexing the scan nadir class (first index of …Num(i,j)) PD.HiIndx.RgAmb(2) = end value for indexing phase breaks in ambiguity number within scans (second index of SC.RgAmb.Num.*Il*(isnc,j)) To support a transport independent data model, the range of the index for any variable sized array is made available as a metadata item. This enables us to have the same form of array support for range ambiguity data as any other lidar metadata variable sized array. <u>Default when not populated:</u> If PD.HiIndx.RgAmb(j) is not populated, the ambiguity state of the data collection is not documented. No references to SC.RgAmb.Num.*Il*(isnc,j) should be made by an application that processes the current data set.
Governing quality metrics	Correct values populated
See Also	SC.RgAmb.Num.*Il*(isnc,j) PF.Subswath.ScansList.*iswl*(isw,it,j)

20.4.53 PD.LoIndx.RgAmb(j) [L1,2,3,4,5]

Brief Definition	PD.LoIndx.RgAmb(j) provides the low value of the indices for the array, SC.RgAmb.Num(i, j).
indices	j=1 start value for indexing the scan nadir class (first index of …Num(i,j)) j=2 start value for indexing phase breaks in ambiguity number within scans (second index of …Num(i,j))
Units/representation	Dimensionless, UInt8
Treatment/usage	To support a transport indepependent data model, the range of the index for any variable sized array is made available as a metadata item. This enables us to have the same form of array support for range ambiguity data as any other lidar metadata variable sized array. <u>Default when not populated:</u> If PD.LoIndx.RgAmb(j) is not populated, the ambiguity state of the data collection is not documented. No references to SC.RgAmb.Num.*Il*(isnc,j) should be made by an application that processes the current data set.
Governing quality metrics	Correct values populated
See Also	SC.RgAmb.Num.*Il*(isnc,j)

20.4.54 PD.HiIndx.Scan (j) [L1,2,3,4,5]

Brief Definition	PD.HiIndx.Scan(j) provides end indices for scan level variable length arrays
indices	j= 1 to 6
Units/representation	Dimensionless, UInt16
Treatment/usage	j=1 end index value for covariance snapshots j=2 end index value for SC.Scan.[N_Major(j,k), Axes.ThetaMaj(j), Axes.ThetaMin(j), ConePulseTable(j), U_Cone(j,k), .NomGroundLen(j)]. When PD.LoIndx.Scan(2) is set to 1, this is the number of scan cone parameter sets for a collection. j=3 end index value for default index of SC.Scan.NadirClass.*ChNum*(is). If PD.LoIndx.Scan(3)=1, then PD.HiIndx.Scan(3) is the total number of scans in the data set. j=4 This is the ending value of the index in the list of invocations of scan parameterizations. If each scan has a different scan cone axis than the preceding scan, PD.HiIndx.Scan(4) will have the value PD.Scan.TotNum, the number of scans in the data set. When scans have repeated cone parameters in consecutive groups, the invocation index increments only on changes and the upper index value is the number of invocations of a scan parameterization for a sequence of scans when PD.LoIndx.Scan(4) is set to 1. j=5 This is the ending value of the index for scan level correlation model

parameters. This gives the number of parameters in the correlation models when PD.LoIndx.Scan(5) is set to 1 and currently should be set to PD.LoIndx.Scan(5) +4 .
j=6 end index value for the list of scan level positions for which geometric position adjustment shifts are supplied. The same number of shifts is provided for all scans in a dataset.

Default when not populated:
If PD.HiIndx.Scan(j) is not populated, scan related metadata is not supported by the data set. Since independent items such as the number of covariance snapshots, scan cone parameter sets and nadir class descriptions of lidar frames are covered in this array, we need to state the following regarding supplying only part of the array: Any item which is not supplied when at least one of the 4 items is supplied will use the value, 65535 to indicate a N/A value.

Governing quality metrics

See Also: SC.Scan.NomGroundLen(j) SC.Scan.ConePulseTable(j), ES.Scan.SigmaX(j) ES.Scan.CorrParms.Alpha(j)

20.4.55 PD.LoIndx.Scan(j) [L1,2,3,4,5]

Brief Definition: PD.LoIndx.Scan(j) provides start indices for scan level variable length arrays

indices: j= 1 to 6

Units/representation: Dimensionless, UInt8

Treatment/usage:
j=1 start index for covariance snapshots found in ES.Scan.*
j=2 starting index value for counting parameterizations for all scan cone parameters such as SC.Scan.N_Major(iscp,ich,k). When PD.LoIndx.Scan(2)=1, PD.HiIndx.Scan(2) will be the number of scan parameterizations for the data set.
j=3 start index for default index of SC.Scan.NadirClass.*ChNum*(is). If PD.LoIndx.Scan(3)=1, then PD.HiIndx.Scan(3) is the total number of scans in the data set.
j=4 This is the starting value of the index in list of a scan cone parameterization invocations. Whenever a pulse is identified as being the first pulse of a run of pulses using a particular set of scan cone parameter values, this is regarded as an invocation. PD.LoIndx.Scan(4) is the index of the first invocation of a scan cone parameterization set. The index increments only on changes of the scan cone parameterization choice. If PD.LoIndx.Scan(4) is set to 1, PD.HiIndx.Scan(4) will be the number of invocations.
j=5 This is the starting value of the index for scan level correlation model parameters. When PD.LoIndx.Scan(5) is set to 1, PD.HiIndx.Scan(5) gives the number parameters in the correlation models. PD.LoIndx.Scan(5) is

typically set to either 0 or 1 depending on the conventions of the implementing language and coding style.
j=6 starting index value for the list of scan level positions for which geometric position adjustment shifts are supplied

Default when not populated:
If PD.LoIndx.Scan(j) is not populated, scan related metadata is not supported by the data set. Since independent items such as the number of covariance snapshots, scan cone parameter sets and nadir class descriptions of lidar frames are covered in this array, we need to state the following regarding supplying only part of the array: If any item is intended for population, all 4 elements of the array will be set to 0 or 1 depending on the implementer's choice of starting index value.

Governing quality metrics

See Also: SC.Scan.NomGroundLen(j) SC.Scan.ConePulseTable(j), ES.Scan.SigmaX(j) ES.Scan.CorrParms.Alpha(j)

20.4.56 PD.HiIndx.SubSwAdj(iswl) [L2,3,4,5]

Brief Definition: PD.HiIndx.SubSwAdj(iswl) provides the end index value for lists of adjustments at subswath level iswl.

indices: Index, iswl, ranges from 1 to PF.Subswath.Nlayers, that is, over the number of subswath layers created in (parallel) processing the data.

Units/representation: dimensionless, UInt8

Treatment/usage: Lists of subswath level adjustment positions are available via PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). Index limits for those lists are provided via PD.LoIndx.SubSwAdj(iswl) and PD.HiIndx.SubSwAdj(iswl).
Default when not populated:
Without PD.HiIndx.SubSwAdj(iswl) there is no support for providing spatially variable geometric corrections at the subswath level via PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). At best, only mean corrections at the subswath level could be populated in the dataset. However, adjustment data for scans can be provided when subswath adjustment data are not populated, permitting sub-subswath scale adjustments to be provided even when internal adjustments at the subswath level are not provided. The impact of not providing subswath level adjustments is that distribution of geometric error may be less uniform than might be otherwise supportable, especially when a single dataset is the only source of registration based adjustments.

Governing quality

metrics	
See Also	PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j)

20.4.57 PD.LoIndx.SubSwAdj(iswl) [L2,3,4,5]

Definition	PD.LoIndx.SubSwAdj(iswl) provides the starting index value for lists of adjustments at subswath level iswl.
indices	Index, iswl, ranges from 1 to PF.Subswath.Nlayers, that is, over the number of subswath layers created in (parallel) processing the data.
Units/representation	dimensionless, UInt8
Treatment/usage	Lists of subswath level adjustment positions are available via PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j). Index limits for those lists are provided via PD.LoIndx.SubSwAdj(iswl) and PD.HiIndx.SubSwAdj(iswl). Default when not populated: See discussion in PD.HiIndx.SubSwAdj(iswl).
Governing quality metrics	
See Also	PF.Adjustmts.Diffs.Positions.Subswath.*Iswl*(isw, i, j)

20.4.58 PD.HiIndx.SubSwScGrpClasses(iswl) [L2,3,4,5]

Brief Definition	PD.HiIndx.SubSwScGrpClasses(iswl) supports the array index range of "it" in PF.SubSwath.ScansList.*iswl*(isw,it,j) with maximum range values.
indices	Index, iswl, ranges from 1 to PF.Subswath.Nlayers, that is, over the number of subswath layers created in (parallel) processing the data.
Units/representation	dimensionless, UInt8
Treatment/usage	PD.HiIndx.SubSwScGrpClasses(iswl) is part of the metadata scheme for describing subswaths. The subswaths are geographically convex regions[174] that are defined in terms of the scans that are used to compose them. The spatial coverage of a scan depends on the time interval[175] in which it was collected and, in the case of spatial channelization, additionally what the channel number was for the scan. A group of scans for a geographical region collected into a subswath with common channel and scan sequence is called a subswath scan group. A subswath scan group is uniquely identified by a value of index "it", and the

[174] That is, they have a convex boundary, but may not necessarily be filled with data points.

[175] This selects the scan parameter values which drive the projection of the sensor instantaneous FOV to the earth frame.

channel is an attribute of "it".[176] For another value of "it", the channel property of the scan group could be different. Another property of the subswath scan group is the aspect (that is, the look direction cone) interval from which the scene points were observed. The combination of channel and aspect vary between separate values of "it"

When PD.LoIndx.SubSwScGrpClasses(iswl)=1, PD.HiIndx.SubSwScGrpClasses(iswl) is the number of temporally consecutive[177] scan sequences in a subswath at subswath layer iswl.

This index control parameter enables subswaths to be defined for modes that involve revisiting specific scene regions to achieve geometric diversity within a single collection operation. In modes that do not employ revisiting a scene position at multiple times within a single collection operation, PD.LoIndx.SubSwScGrpClasses(iswl)=PD.HiIndx.SubSwScGrpClasses(iswl).

Default when not populated:
If PD.HiIndx.SubSwScGrpClasses(iswl) is not populated, there is no mapping of scans to subswaths and PF.SubSwath.ScansList.*iswl*(isw,it,j) should not be referenced by an application that processes the current data set.

Governing quality metrics

See Also PF.SubSwath.ScansList.*iswl*(isw,it,j)

20.4.59 PD.LoIndx.SubSwScGrpClasses(iswl) [L2,3,4,5]

Definition PD.LoIndx.SubSwScGrpClasses(iswl) supports the array index range of "it" in PF.SubSwath.ScansList.*iswl*(isw,it,j) with minimum range values.

indices Index, iswl, ranges from 1 to PF.Subswath.Nlayers, that is, over the number of subswath layers created in (parallel) processing the data.

Units/representation dimensionless, UInt8

Treatment/usage When PD.LoIndx.SubSwScGrpClasses(iswl)=1, PD.HiIndx.SubSwScGrpClasses(iswl) is the number of temporally

[176] Note that this discussion is constrained to a specific subswath. The same value of "it" could contribute to a different subswath, but that would have a different channel value. Without the subswath constraint, it is not true that channel is an attribute of "it".

[177] Meaning that each sequence of scans is internally ordered by time with no "gaps", but the temporal series of the sequence's start times may have gaps longer than a scan sequence. That is, a subswath could correspond to collection over separated time intervals in which the sensor was used during a gap to collect a region in a different subswath.

consecutive scan sequences in a subswath at subswath layer iswl.

Default when not populated:
If PD.LoIndx.SubSwScGrpClasses(iswl) is not populated, there is no mapping of scans to subswaths and PF.SubSwath.ScansList.*iswl*(isw,it,j) should not be referenced by an application that processes the current data set.

Governing quality metrics

See Also PF.SubSwath.ScansList.*iswl*(isw,it,j) PD.HiIndx.SubSwScGrpClasses(iswl)

20.4.60 PD.HiIndx.Swath(j) [L1,2,3,4,5]

Brief Definition PD.HiIndx.Swath(j) provides end index values for swath level variable length arrays

indices Index, j, ranges from 1 to 13

Units/representation Dimensionless, UInt16

Treatment/usage PD.HiIndx.Swath(j) provides index range information for "swath level" data. By "swath level", we mean either items that are fixed for the entire collection operation or items that due to processing (for example, items that feel the impact of registrations) represent versions that apply to the full product.
j=1 end index value for covariance snapshots. This is the ending value of the first index in swath level covariance data including correlation model parameters. Governs ES.Swath.SigmaX(k), ES.Swath.CorrParms.X(j,k) etc.
j=2 end index value for row index of beam illumination pattern. Governs SC.Laser.Beam(il,**i**,j)
j=3 end index value for column index of beam illumination pattern. Governs SC.Laser.Beam(il,i,**j**)
j=4 end index value of iv in the MD.Coverage.Bounds.Vertices(iv,j) data array
j=5 end index value of j in the MD.Coverage.CountryCode(j) data array. When PD.LoIndx.Swath(5)=1, PD.HiIndx.Swath(5) is the number of countries in the country code list for the product.
j=6 end index value of j in MD.Coverage.BE_NumList(j)
j=7 end index value of iv in MD.Objective.Vertices(iv,j)
j=8 end index value of j in MD.Objective.WAC_List(j,k)
j=9 end index value of j in MD.Objective.BE_NumList(j)
j=10 end index value of j in SC.MultiDwell.Dat(j) currently set to PD.LoIndx.Swath(10) +2
j=11 end index value of j in PD.HiIndx.IPP(j)
j=12 end index value for correlation model parameters. This is the ending value of the index for swath level correlation model parameters. This gives

the number of parameters in the correlation models when PD.LoIndx.Swath(12) is set to 1 and currently should be set to PD.LoIndx.Swath(12) +4 .
j=13 end index value for the list of conversion factors between standard deviations and .95P values

Default when not populated:
If we populate the PD.HiIndx.Swath(j) array in order to support some but not all of the items whose upper bound index ranges are listed above, then any N/A item should be given the value 65535. If the PD.HiIndx.Swath(j) array is not populated at all, there is no support for a range of information items about the swath level product. All items would take on the N/A coded default value of 65535. Any item given a value of 65535 corresponds to another metadata item that is not populated and those corresponding items should not be referenced by an application using the current data set.

Governing quality metrics

See Also: ES.Swath.SigmaY(k), ES.Swath.CorrParms.Y(k) SC.Laser.Beam(il,**i**,j) MD.Coverage.CntryCd2(j) SC.MultiDwell.Dat(j) PD.HiIndx.IPP(j)

20.4.61 PD.LoIndx.Swath(j) [L1,2,3,4,5]

Brief Definition: PD.LoIndx.Swath(j) provides start index values for swath level variable length arrays.

indices: Index, j, ranges from 1 to 13

Units/representation: Dimensionless, UInt8

Treatment/usage:
j=1 start index for covariance snapshots. This is the starting value of the first index in swath level covariance data including correlation model parameters. Governs ES.Swath.SigmaX(k), ES.Swath.CorrParms.X(j,k) etc.
j=2 start index for row index of beam illumination pattern. Governs SC.Laser.Beam(il,**i**,j)
j=3 start index for column index of beam illumination pattern. Governs SC.Laser.Beam(il,i,**j**)
j=4 start index value of iv in the MD.Coverage.Bounds.Vertices(iv,j) data array
j=5 start index value of j in the MD.Coverage.CountryCode(j) data array
j=6 start index value of j in MD.Coverage.BE_NumList(j)
j=7 start index value of iv in MD.Objective.Vertices(iv,j)
j=8 start index value of j in MD.Objective.WAC_List(j,k)
j=9 start index value of j in MD.Objective.BE_NumList(j)
j=10 start index value of j in SC.MultiDwell.Dat(j) If this is set to 1, PD.HiIndx.Swath(10) will be the number of parameters defining a dwell in a

multi-dwell collect.
j=11 start index value of j in PD.HiIndx.IPP(j)
j=12 start index value for correlation model parameters. When PD.LoIndx.Swath(12) is set to 1, PD.HiIndx.Swath(12) gives the number of parameters in the correlation models. PD.LoIndx.Swath(12) is typically set to either 0 or 1 depending on the conventions of the implementing language and coding style.
j=13 start index value for the list of conversion factors between standard deviations and .95P values

Default when not populated:
If we populate the PD.LoIndx.Swath(j) array in order to support some but not all of the items whose upper bound index ranges are listed above, then any N/A item should be given the value 65535. If the PD.LoIndx.Swath(j) array is not populated at all, there is no support for a range of information items about the swath level product. All items would take on the N/A coded default value of 65535. Any item given a value of 65535 corresponds to another metadata item that is not populated and those corresponding items should not be referenced by an application using the current data set. Giving a N/A value to an entry in PD.LoIndx.Swath(j) or the corresponding entry in PD.HiIndx.Swath(j) indicates that the corresponding metadata array whose index ranges are given by PD.LoIndx.Swath(j) and PD.HiIndx.Swath(j) should go unreferenced by an application using the current data set.

Governing quality metrics

See Also: ES.Swath.SigmaY(k), ES.Swath.CorrParms.Y(k) PD.HiIndx.Swath(j) PD.HiIndx.IPP(j)

20.4.62 PD.HiIndx.TileMap.DedictdTile [L1,2,3,4,5]

Brief Definition: PD.HiIndx.TileMap.DedictdTile gives the ending value for the index of dedicated tile description arrays.

indices: none

Units/representation: Dimensionless UInt16

Treatment/usage: PD.HiIndx.TileMap.DedictdTile is the index of the last tile group dedicated to a single value of some parameters in the data row.
If PD.LoIndx.TileMap.DedictdTile were set to 1, then PD.HiIndx.TileMap.DedictdTile would be set to the number of tile groups with a single value for each of a subset of the data row parameters. Governs
PD.TileMap.DedictdTile.Cnstrnt1.Val(j),
PD.TileMap.DedictdTile.Cnstrnt2.Val(j),
PD.TileMap.DedictdTile.Cnstrnt3.Val(j),

PD.TileMap.DedictdTile.TileNumValLow(j),
PD.TileMap.DedictdTile.TileNumValHi(j)
A product employing a dedicated tile strategy would have a table such as that shown in section 10.2.1.5. There is one row in this table for each tile group.

Default when not populated:
If PD.HiIndx.TileMap.DedictdTile is not populated, dedicated tile strategy is not in use for the current data set. PD.TileMap.DedictdTile.Cnstrnt1.Val(j), PD.TileMap.DedictdTile.Cnstrnt2.Val(j), PD.TileMap.DedictdTile.Cnstrnt3.Val(j), PD.TileMap.DedictdTile.TileNumValLow(j), and PD.TileMap.DedictdTile.TileNumValHi(j) should not be referenced.

Governing quality metrics

See Also PD.TileMap.DedictdTile.Cnstrnt1.Val(j), PD.TileMap.DedictdTile.TileNumValHi(j)

20.4.63 PD.LoIndx.TileMap.DedictdTile [L1,2,3,4,5]

Brief Definition PD.LoIndx.TileMap.DedictdTile gives the starting value for the index of dedicated tile description arrays.

indices none

Units/representation Dimensionless UInt8

Treatment/usage This is the index of the first tile group dedicated to a single value of some parameters in the data row (see figure 10.1.2-4).
If PD.LoIndx.TileMap.DedictdTile were set to 1, then PD.HiIndx.TileMap.DedictdTile would be set to the number of tile groups with a single value for each of a subset of the data row parameters.
Governs PD.TileMap.DedictdTile.Cnstrnt1.Val(j),
PD.TileMap.DedictdTile.Cnstrnt2.Val(j),
PD.TileMap.DedictdTile.Cnstrnt3.Val(j),
PD.TileMap.DedictdTile.TileNumValLow(j),
PD.TileMap.DedictdTile.TileNumValHi(j)
Dedicated tiles are an optional data set design strategy and the dedicated tile descriptors need not be populated if no tile is dedicated to a single value of some parameter.

Default when not populated:
If PD.LoIndx.TileMap.DedictdTile is not populated, dedicated tile strategy is not in use for the current data set. PD.TileMap.DedictdTile.Cnstrnt1.Val(j),
PD.TileMap.DedictdTile.Cnstrnt2.Val(j),
PD.TileMap.DedictdTile.Cnstrnt3.Val(j),

PD.TileMap.DedictdTile.TileNumValLow(j), and
PD.TileMap.DedictdTile.TileNumValHi(j) should not be referenced.

Governing quality metrics

See Also

20.4.64 PD.HiIndx.Tsegts.*TileNum* [L2,3,4,5]

Brief Definition: PD.HiIndx.Tsegts.*TileNum* gives the last value of the range for the second index of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)

indices: None
However, the *TileNum* is an evaluated sub-string of the tagname that acts somewhat like and index. The value of *TileNum* must be kept in correspondence with the value of *TileNum* in the instantiation of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) for which PD.HiIndx.Tsegts.*TileNum* applies.

Units/representation: Dimensionless, UInt8

Treatment/usage: When PD.LoIndx.Tsegts.*TileNum* is set to 1, PD.HiIndx.Tsegts.*TileNum* will be the number of intra-tile time segments employed in ES.Point.PulseNum.*TileNum*(ipt).

Default when not populated:
If PD.HiIndx.Tsegts.*TileNum* is not populated, there will be no intra-tile time segments in use and ES.Point.PulseNum.*TileNum*(ipt) should not be referenced by an application using the current data set as source data.

Governing quality metrics

See Also: PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)

20.4.65 PD.LoIndx.Tsegts.*TileNum* [L2,3,4,5]

Brief Definition: PD.LoIndx.Tsegts.*TileNum* gives the first value of the range for the second index of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)

indices: None
However, the *TileNum* is an evaluated sub-string of the tagname that acts somewhat like and index. The value of *TileNum* must be kept in correspondence with the value of *TileNum* in the instantiation of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) for which PD.HiIndx.Tsegts.*TileNum* applies.

Units/representation: Dimensionless, UInt8

Treatment/usage: When PD.LoIndx.Tsegts.*TileNum* is set to 1, PD.HiIndx.Tsegts.*TileNum* will

be the number of intra-tile time segments employed in ES.Point.PulseNum.*TileNum*(ipt). PD.LoIndx.Tsegts.*TileNum* will generally be set to 1 or 0, depending on the coding strategy of the processor application.

Default when not populated:
If PD.LoIndx.Tsegts.*TileNum* is not populated, there will be no intra-tile time segments in use and ES.Point.PulseNum.*TileNum*(ipt) should not be referenced by an application using the current data set as source data.

Governing quality metrics

See Also PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)

20.4.66 PD.HiIndx.Truth(j) [L1,2,3,4,5]

Brief Definition PD.HiIndx.Truth(j) gives the last value of indices in ground truth related data items.

indices Index j=1,2

Units/representation Dimensionless, UInt8

Treatment/usage PD.HiIndx.Truth(1) = the upper value of itp, the truth point index. When PD.LoIndx.Truth(1) is given to be 1, PD.HiIndx.Truth(1) is the number of truth points.
PD.HiIndx.Truth(2) = the upper value of j, the truth point illustration index. See PD.Wordsize.Truth(j). When PD.LoIndx.Truth(2) is given to be 1, PD.HiIndx.Truth(2) is the number of truth point illustrations. Ideally, an illustration would be provided for each local truthed scene region resulting in the PD.Lo/HiIndx.Truth ranges being the same in practice.

Default when not populated:
If PD.HiIndx.Truth(j) is not populated, ground truth data will not be present in the current data set. Es.Truth.Pos(itp,j) should not be referenced by an application using the current data set as input.

Governing quality metrics

See Also ES.Truth.Pos(itp,j) PD.Wordsize.Truth(j)

20.4.67 PD.LoIndx.Truth(j) [L1,2,3,4,5]

Brief Definition PD.LoIndx.Truth(j) gives the first value of indices in ground truth related data items.

indices Index j=1,2

Units/representation Dimensionless, UInt8

Treatment/usage PD.LoIndx.Truth(1) = the lower value of itp, the truth point index. When PD.LoIndx.Truth(1) is given to be 1, PD.HiIndx.Truth(1) is the number of

truth points.
PD.HiIndx.Truth(2) = the lower value of j, the truth point illustration index. See PD.Wordsize.Truth(j). When PD.LoIndx.Truth(2) is given to be 1, PD.HiIndx.Truth(2) is the number of truth point illustrations.

Default when not populated:
If PD.LoIndx.Truth(j) is not populated, ground truth data will not be present in the current data set. Es.Truth.Pos(itp,j) should not be referenced by an application using the current data set as input.

Governing quality metrics

See Also: ES.Truth.Pos(itp,j) PD.Wordsize.Truth(j)

20.4.68 PD.HiIndx.UDNB*n* [L1,2,3,4,5]

Brief Definition: PD.HiIndx.UDNB*n* gives the last value of the sequence of the j index for user defined narrow band data item PF.UserDefined.NB_Dat.P*n*.*TileNum*(j)

indices: None
The evaluated sub-string "*n*", however, behaves somewhat as an index. A separate value of *n* is used for each user defined narrow band parameter. So, in the case where a user has definitions for a first and second narrow band parameter, the data set would contain two narrow band data items. These might be, for example, PF.UserDefined.NB_Dat.P1.*TileNum*(j) and PF.UserDefined.NB_Dat.P2.*TileNum*(j). These could be different length data vectors.

Units/representation: Dimensionless, UInt16

Treatment/usage: Per 10.4, this CMMD supports user defined narrow band data through the trio, PF.UserDefined.NB_Dat.P*n*.*TileNum*(j), and its descriptors, PD.UserDefined.NB_Def.P*n* and PD.UserDefined.NB_Type.P*n*. The data item may appear as an array. To provide general support, the range of the index j is provided in PD.Lo/HiIndx.UDNB*n*. A single data item, that is a vector of length 1, is documented by giving the same value to both PD.LoIndx.UDNB*n* and PD.HiIndx.UDNB*n*.

Default when not populated:
If PD.HiIndx.UDNB*n* is not populated, no user defined narrowband metadata items are present and PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) should not be referenced by an application using the current data set.

Governing quality metrics: Must be the correct value.

See Also: PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) PD.UserDefined.NB_Type.P*n*

20.4.69 PD.LoIndx.UDNB*n* [L1,2,3,4,5]

Brief Definition	PD.LoIndx.UDNB*n* gives the first value of the sequence of the j index for user defined narrow band data item PF.UserDefined.NB_Dat.P*n*.*TileNum*(j)
indices	None The evaluated sub-string "*n*", however, behaves somewhat as an index. A separate value of *n* is used for each user defined narrow band parameter. So, in the case where a user has definitions for a first and second narrow band parameter, the data set would contain two narrow band data items. These might be, for example, PF.UserDefined.NB_Dat.P1.*TileNum*(j) and PF.UserDefined.NB_Dat.P2.*TileNum*(j). These could be different length data vectors.
Units/representation	Dimensionless, UInt16
Treatment/usage	Per 10.4, this CMMD supports user defined narrow band data through the trio, PF.UserDefined.NB_Dat.P*n*.*TileNum*(j), and its descriptors, PD.UserDefined.NB_Def.P*n* and PD.UserDefined.NB_Type.P*n*(j). The data item may appear as an array. To provide general support, the range of the index j is provided in PD.Lo/HiIndx.UDNB*n*. A single data item, that is a vector of length 1, is documented by giving the same value to both PD.LoIndx.UDNB*n* and PD.HiIndx.UDNB*n*. Default when not populated: If PD.LoIndx.UDNB*n* is not populated, no user defined narrowband metadata items are present and PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) should not be referenced by an application using the current data set.
Governing quality metrics	Must be the correct value.
See Also	PF.UserDefined.NB_Dat.P*n*.*TileNum*(j)

20.4.70 PD.HiIndx.Voids(j) [L1,2,3]

Brief Definition	PD.HiIndx.Voids(j) gives the last value of the void index for each of the sequences of Void data in MD.Coverage.Voids.*
indices	Index, j, =1,2,3,4
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.HiIndx.Voids(1)= ending value of index j in MD.Coverage.Voids.Atmsphrc(j,k). When PD.LoIndx.Voids(1)=1, PD.HiIndx.Voids(1) is the number of reported atmospherically caused voids described in MD.Coverage.Voids.Atmsphrc(j,k). PD.HiIndx.Voids(2)= ending value of index j in MD.Coverage.Voids.Incompl(j,k).

When PD.LoIndx.Voids(2)=1, PD.HiIndx.Voids(2) is the number of reported incomplete collection voids described in MD.Coverage.Voids.Incompl(j,k).

PD.HiIndx.Voids(3)= ending value of index j in MD.Coverage.Voids.Masked (j,k). When PD.LoIndx.Voids(3)=1, PD.HiIndx.Voids(3) is the number of reported terrain masking voids described in MD.Coverage.Voids.Masked(j,k).

PD.HiIndx.Voids(4)=ending value of index j in MD.Coverage.Voids.UnColl(j,k). When PD.LoIndx.Voids(4)=1, PD.HiIndx.Voids(4) is the number of reported uncollected area voids described in MD.Coverage.Voids.UnColl(j,k).

Default when not populated:
If at least one element of PD.HiIndx.Voids(j) is to be populated meaningfully, any N/A elements will be given the value, 255. If the PD.HiIndx.Voids(j) array is not populated at all, that is equivalent to each element having the value 255 and the coverage data items corresponding to the N/A index range data should not be referenced by an application using the current data set.

Governing quality metrics: Must be the correct value.

See Also: MD.Coverage.Voids.Atmsphrc(j,k) MD.Coverage.Voids.Incompl(j,k) MD.Coverage.Voids.Masked(j,k) MD.Coverage.Voids.UnColl(j,k)

20.4.71 PD.LoIndx.Voids(j) [L1,2,3]

Brief Definition: PD.LoIndx.Voids(j) gives the first value of the void index for each of the sequences of Void data in MD.Coverage.Voids.*

indices: Index, j, =1,2,3,4

Units/representation: Dimensionless, UInt8

Treatment/usage:
PD.LoIndx.Voids(1)=starting value of index j in MD.Coverage.Voids.Atmsphrc(j,k).
PD.LoIndx.Voids(2)=starting value of index j in MD.Coverage.Voids.Incompl(j,k).
PD.LoIndx.Voids(3)=starting value of index j in MD.Coverage.Voids.Masked (j,k).
PD.LoIndx.Voids(4)=starting value of index j in MD.Coverage.Voids.UnColl(j,k).

Default when not populated:
If at least one element of PD.LoIndx.Voids(j) is to be populated meaningfully, any N/A elements will be given the value, 255. If the PD.LoIndx.Voids(j) array is not populated at all, that is equivalent to each element having the value 255 and the coverage data items corresponding to the N/A index range data should not be referenced by an application using the current data set.

Governing quality metrics: Must be the correct value.

See Also:

20.4.72 PD.HiIndx.WF(j) [L1,2]

Brief Definition	PD.HiIndx.WF(j) gives the last value of the sequence of the j_tile index for tiles associated with waveform data and the uppler limit on waveform return numbers.
indices	Index, j, =1,2
Units/representation	Dimensionless, UInt16
Treatment/usage	In the list that associates an integer index to each evaluation of *TileNum*, there may be a subsequence for waveform related data. j=1---PD.HiIndx.WF(1) is the highest value of j_tile in PD.TileMap.TileNum(j_tile) associated with waveform data bearing tiles. Used to govern PD.TileMap.RayRange(j_tile,k). j=2---PD.HiIndx.WF(2) is the highest value of the second index of PD.TileMap.WF_Returns(ich,j), that is, the highest number of wave clip returns on any pulse-ray for a given channel. This will be generally 7 or less. <u>Default when not populated:</u> At enterprise level-1, only PD.HiIndx.WF(1) is applicable and it is <u>essential</u>. Thus PD.HiIndx.WF(2) should be ignored at level-1 and no special code is needed to indicate this is the case. This will be determined by MD.ProcHist.ProdType. If PD.HiIndx.WF(j) is not populated at enterprise level-2, there are no waveform bearing tiles and PD.TileMap.RayRange(j_tile,k) should not be referenced . A plausible coding technique might be to default PD.HiIndx.WF(1) to a value greater than PD.TileMap.N_Tiles and trapping on that condition to indicate the absence of waveform data in a (L2) data set.
Governing quality metrics	Must be the correct value.
See Also	PD.TileMap.TileNum(j_tile) PD.TileMap.WF_Returns(ich,j)

20.4.73 PD.LoIndx.WF(j) [L1,2]

Brief Definition	PD.LoIndx.WF(j) gives the first value of the sequence of the j_tile index for tiles associated with waveform data.
indices	Index, j, =1,2
Units/representation	Dimensionless, UInt8
Treatment/usage	In the list that associates an integer index to each evaluation of *TileNum*, there may be a subsequence for waveform related data. j=1---PD.LoIndx.WF(1) is the lowest value of j_tile in PD.TileMap.TileNum(j_tile) associated with waveform data bearing tiles. Used to govern PD.TileMap.RayRange(j_tile,k). j=2---PD.LoIndx.WF(2) is the lowest value of the second index of

PD.TileMap.WF_Returns(ia,j), that is, the lowest number of wave clip returns on any pulse ray for a given array. If PD.LoIndx.WF(2) is set to 1, PD.HiIndx.WF(2) will be the number of returns on the pulse-ray with the most return waveform clips.

Default when not populated:
At enterprise level-1, only PD.LoIndx.WF(1) is applicable and it is essential. Thus PD.LoIndx.WF(2) should be ignored at level-1 and no special code is needed to indicate this is the case. This will be determined by MD.ProcHist.ProdType. If PD.LoIndx.WF(j) is not populated at enterprise level-2, there are no waveform bearing tiles and PD.TileMap.RayRange(j_tile,k) should not be referenced . A plausible coding technique might be to default PD.LoIndx.WF(1) to a value greater than PD.TileMap.N_Tiles and trapping on that condition to indicate the absence of waveform data in a (L2) data set.

Governing quality metrics: Must be the correct value.

See Also: PD.TileMap.TileNum(j_tile)

20.4.74 PD.HiIndx.WFClpPulseNum.*TileNum* [L2]

Brief Definition: PD.HiIndx.WFClpPulseNum.*TileNum* gives the last value of the index for the sequence of pulses used in a specific tile of waveform clip data.

indices: none

Units/representation: Dimensionless, UInt32

Treatment/usage: Particularly for lidar enterprise level-2 data sets, the number of waveform clips provided may be far less than the number of pulses used in the data collection operation. To support efficient byte volume implementations, we introduce the index, ipi, which ranges over only the pulses that actually contribute waveform clips to each return specific tile.
PD.HiIndx.WFClpPulseNum.*TileNum* provides the high index value for the range of ipi as in PD.WFClp.PulseNum.*TileNum*(ipi), ES.WFClp.Locs.*TileNum*(ipi,iray,j).
If PD.LoIndx.WFClpPulseNum.*TileNum* is set to 1, PD.HiIndx.WFClpPulseNum.*TileNum* will be the number of "used pulses" in sparse wave clip tile, *TileNum*.

Default when not populated:
If PD.HiIndx.WFClpPulseNum.*TileNum* is not populated, there will be no “used pulses only” strategy employed in populating the received waveform recorded data and PD.WFClp.PulseNum.*TileNum*(ipi) should not be referenced by an application using the current data set as source data.

Governing quality: Must be the correct value.

metrics	
See Also	ES.WFClp.Locs.*TileNum*(ipi,iray,j) PD.WFClp.PulseNum.*TileNum*(ipi)

20.4.75 PD.LoIndx.WFClpPulseNum.*TileNum*[L2]

Brief Definition	PD.LoIndx.WFClpPulseNum.*TileNum* gives the first value of the index for the sequence of pulses used in a specific tile waveform clip data.
indices	none
Units/representation	Dimensionless, UInt32
Treatment/usage	Particularly for lidar enterprise level-2 data sets, the number of waveform clips provided may be far less than the number of pulses used in the data collection operation. To support efficient byte volume implementations, we introduce the index, ipi, which ranges over only the pulses that actually contribute waveform clips to each return specific tile. PD.LoIndx.WF_PulseNum.*TileNum* provides the low index value for the range of ipi as in PD.WF_PulseNum.*TileNum*(ipi) , ES.WFClp.Locs.*TileNum*(ipi,iray,j). If PD.LoIndx.WF_PulseNum.*TileNum* is set to 1, PD.HiIndx.WF_PulseNum.*TileNum* will be the number of "used pulses" in sparse wave clip tile, *TileNum*. PD.LoIndx.WF_PulseNum.*TileNum* is generally set to 1 or 0 per the implementer's choice. <u>Default when not populated:</u> If PD.LoIndx.WFClpPulseNum.*TileNum* is not populated, there will be no "used pulses only" strategy employed in populating the received waveform recorded data and PD.WFClp.PulseNum.*TileNum*(ipi) should not be referenced by an application using the current data set as source data.
Governing quality metrics	Must be the correct value.
See Also	PD.HiIndx.WF_PulseNum.*TileNum*

20.4.76 PD.HiIndx.Zones [L1,2,3,4,5]

Brief Definition	PD.HiIndx.Zones gives the last value of the index, j, of MD.Coverage.Bounds.ZoneList(j)
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	When PD.LoIndx.Zones is set to 1, PD.HiIndx.Zones will be the number of zones listed as covering the current data set. Note that the size of a zone depends on the number of characters in the designator. Thus, a fixed data set geographic coverage can have a variety of values for the number of zones to

cover it depending on the zone sizes, which are driven by the value of PD.Wordsize.Zones.
Inheritance of this value from enterprise level-1 to level-2 is not necessarily cut and paste. It is highly likely that at level-1 a large zone increment would be used, but that the same coverage at level-2 could employ a substantially larger number of smaller zones. If level-3 chips are created as new files, the number and size of the zones may both be reduced relative to the source data set.

Default when not populated:
If PD.HiIndx.Zones is not populated, a list of MGRS zones covered by the data set is not provided and applications using the current data set as input should not find a usable MD.Coverage.Bounds.ZoneList(j) within.

Governing quality metrics: Must be the correct value.

See Also: MD.Coverage.Bounds.ZoneList(j) PD.Wordsize.Zones

20.4.77 PD.LoIndx.Zones [L1,2,3,4,5]

Brief Definition: PD.LoIndx.Zones gives the first value of the lowest value of the index, j, of MD.Coverage.Bounds.ZoneList(j)

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: This is determined by the writers of the lidar data set generating code and will generally be code design driven reflecting the indexing conventions of the language use to write the code. Generally, this will be set to either 0 or 1. When PD.LoIndx.Zones is set to 1, PD.HiIndx.Zones will be the number of zones listed as covering the current data set.

Default when not populated:
If PD.LoIndx.Zones is not populated, a list of MGRS zones covered by the data set is not provided and applications using the current data set as input should not find a usable MD.Coverage.Bounds.ZoneList(j) within.

Governing quality metrics: Must be the correct value.

See Also: MD.Coverage.Bounds.ZoneList(j)

PD.Point.*

PD.Point.* data items define index ranges for applicability of ES.* error data.

20.4.78 PD.Point.RangeListC.*OrdIndx*(j,k) [L2,3,4,5]

Brief Definition

PD.Point.RangeListC.*OrdIndx*(j,k) is a list of point index ranges for which the j1=*OrdIndx* error ellipsoid in ES.Point.SigmaXYZ(j1,k) is the default common value.

indices

The substring, *OrdIndx* is an evaluated piece of the tag name that is a character representation of a value of the j index of ES.Point.SigmaXYZ(j,k). As such, *OrdIndx* acts much like an index, even though it is part of the parameter name.

j ranges from PD.LoIndx.Point.RangeListC.*OrdIndx* to PD.HiIndx.Point.RangeListC.*OrdIndx*, that is, from the index value of the first range of points using ES.Point.SigmaXYZ(*OrdIndx*,k) as the common covariance value to the index value of the last range of points. If PD.LoIndx.Point.RangeListC.*OrdIndx* were set to 1, PD.HiIndx.Point.RangeListC.*OrdIndx* would be the number of point ranges in the complete data set that have ES.Point.SigmaXYZ(*OrdIndx*,k[178]) as the common value of the covariance matrix.
k=1,2,3

Units/representation

Dimensionless, UInt32

Treatment/usage

While this CMMD can support formal error propagation for any particular point in a point cloud, it may be quite common that earth frame point covariances are effectively the same for a large percentage of the points in a data set. In such a case, it is a great convenience to pre-compute the ground space covariance once for all points for which the value is expected to be approximately the same. The covariance data from which the ellipsoid axes and tilts can be computed are supplied in the array, ES.Point.SigmaXYZ(j,k). PD.Point.RangeListC.*OrdIndx*(j,k) is a list of ranges of points as they are ordered in tiles that have a common covariance matrix.

OrdIndx will always be evaluated to a character string that is BCS code for an integer, such as “1”, “001” or “02”. If *OrdIndx*= “02”, or “002” etc., then PD.Point.RangeListC.*OrdIndx* is a list of applicable point ranges for the second enumerated covariance matrix.

The index, k, covers the 3 parameters that define a point range:

1) the tile number
2) The “point row” (whose index we call “ipt”) of the first point of a

[178] Not the k being defined in this paragraph.

range

3) The "point row" of the last point of a range

The point range includes the first, last and intervening points.
Since the index, j, ranges over the individual point ranges, PD.Point.RangeListC.*OrdIndx*(j,1) is the tile number where a range of lidar points given common covariance matrix value lies. This integer value for tiles is linked to the *TileNum* substrings of tiled data via PD.TileMap.TileNum(j_tile).
PD.Point.RangeListC.*OrdIndx*(j,2) is the starting value of ipt, that is, the "point row" that is the beginning of a point range having a given common value of error covariance.
PD.Point.RangeListC.*OrdIndx*(j,3) is the ending value of ipt, that is the "point row" that is the last of a point range having a given common value of error covariance.

If there were 10 different point sigmas computed for a data set, we would see the list of parameter names PD.Point.RangeListC.01, PD.Point.RangeListC.02 and PD.Point.RangeListC.03 if the first 3 point sigma sets were those of common sigma matrix values. Then the remaining 7 sigmas would be values of exceptions. Exceptions are point sigma values that belong to short isolated sequences of points imbedded in longer sequences of points for which the vast majority of points have a point sigma value equal to a single common value. PD.Point.RangeListC.01(j,k) is a table that has a row (value of j) for each range of points for which the common value of the point covariance matrix is given by ES.Point.SigmaXYZ(1,k). The first range of points to which the common value applies is specified by the tile it appears in, the first "point row" value and the last "point row" for which the common covariance applies (also determined by the ipt value of a data row in the LD.* tile). So the first range of points that has the first common sigma value appears as PD.Point.RangeListC.01(1,k) where k=1,2,3. The next following range of points (if there is one) would have a different common value for the covariance matrix and thus would show up in, for example, PD.Point.RangeListC.02(1,k). But if the next range of points again has the first value of covariance matrix, it will appear in PD.Point.RangeListC.01(2,k) where a different list of tile, first ipt and last ipt will appear. So while k has been be fixed at 1,2,3, since we need to specify a tile number, a first point row and a last point row to define a point range, the number of ranges is a variable and needs to have a Lo/HiIndx provided. We provide these as PD.Lo/HiIndx.Point.RangeListC.*OrdIndx.*

The guidance for classifying an earth frame point location error covariance as effectively equivalent to a common (exemplar) value is:

- Consider comparison of a local computed 3x3 error covariance matrix, **C** to an exemplar error covariance $\mathbf{C_{ex}}$.
- Form the matrix difference of **C** and **Cex**, **Cex-C=ΔC.**
- Evaluate the function NormDet= Det(**ΔC)**/Det**(Cex)** where "Det" denotes determinant. When forming Det(**ΔC),** care must be taken to assure that ΔC is rank 3. The easiest way to do that is to assure that the magnitude of its elements are non-zero by adding "ε" (see below) to any zero element.
- Evaluate the asymmetry correction factor ACF=Max(λ_1,λ_2,λ_3)/[Min((λ_1,λ_2,λ_3)+ε] where (λ_1,λ_2,λ_3) are the magnitudes of the eigenvalues of **ΔC** and "ε" is the the smallest positive value that the **λ**s can take (like the least significant bit)
- Form the metric C_{prox} =ACF×NormDet
- Take the cube root of the square of C_{prox} and call this the covariance metric M_C.
- When M_C<0.1, we can use the **Cex** to substitute for **C.**

Also note, that if PD.Point.RangeListC.*OrdIndx*(j,k) and PD.Point.RangeListX.*OrdIndx*(j,k) were populated previous to sorting any data, these tables need to be repopulated to reflect any changes in the mapping of pre-computed error ellipsoids to the points in the point cloud.

Governing quality metrics

See Also

ES.Point.SigmaXYZ(j,k) PD.TileMap.TileNum(j_tile)
PD.HiIndx.Point.RangeListC.*OrdIndx*

20.4.79 PD.Point.RangeListX.*OrdIndx*(j,k) [L2,3,4,5]

Brief Definition

PD.Point.RangeListX.*OrdIndx*(j,k) is a list of point index ranges for which the j1=*OrdIndx* error ellipsoid in ES.Point.SigmaXYZ(j1,k) is an exceptional value.

indices

The substring, *OrdIndx* is an evaluated piece of the tag name that is a character representation of a value of the j index of ES.Point.SigmaXYZ(j,k). As such, *OrdIndx* acts much like an index, even though it is part of the parameter name.

j ranges from PD.LoIndx.RangeListX.*OrdIndx* to PD.HiIndx.RangeListX.*OrdIndx*, that is, from the index value of the first range of points using ES.Point.SigmaXYZ(*OrdIndx*,k) as an exceptional covariance value to the index value of the last range of points. If PD.LoIndx.RangeListX.*OrdIndx* were set to 1, PD.HiIndx.RangeListX.*OrdIndx* would be the number of point ranges in the

complete data set that have ES.Point.SigmaXYZ(*OrdIndx*,k[178]) as an exceptional value of the covariance matrix.
k=1,2,3

Units/representation Dimensionless, UInt32

Treatment/usage While this CMMD can support formal error propagation for any particular point in a point cloud, it may be quite common that earth frame point covariances are effectively the same for a large percentage of the points in a data set. In such a case, it is a great convenience to pre-compute the ground space covariance once for all points for which the value is expected to be approximately the same. The covariance data from which the ellipsoid axes and tilts can be computed are supplied in the array, ES.Point.SigmaXYZ(j,k). Some short ranges of points imbedded within a range with a predominant value of the covariance that we have called "common" may have values that are exceptions. PD.Point.RangeListX.*OrdIndx*(j,k) is a list of ranges of points as they are ordered in tiles that have a particular exceptional value for the covariance matrix.

OrdIndx will always be evaluated to a character string that is BCS code for an integer, such as "1", "001" or "02". If *OrdIndx*= "04", or "004" etc., then PD.Point.RangeListX.*OrdIndx* is a list of applicable exceptional point ranges for the <u>fourth</u> enumerated covariance matrix.

The index, k, covers the 3 parameters that define a point range:

4) the tile number
5) The "point row" (whose index we call "ipt") of the first point of a range
6) The "point row" of the last point of a range

The point range includes the first, last and intervening points.
Since the index, j, ranges over the individual point ranges,
PD.Point.RangeListX.*OrdIndx*(j,1) is the tile number where a range of lidar points given an exceptional covariance matrix value of the *OrdIndx* th listed covariance matrix lies. This integer value for tiles is linked to the *TileNum* substrings of tiled data via PD.TileMap.TileNum(j_tile).
PD.Point.RangeListX.*OrdIndx*(j,2) is the starting value of ipt, that is the "point row" that is the beginning of a point range having a given exceptional value of error covariance.
PD.Point.RangeListX.*OrdIndx*(j,3) is the ending value of ipt, that is the "point row" that is the last of a point range having a given exceptional value of error covariance.

If there were 10 different point sigmas computed for a data set, we would see

the list of parameter names PD.Point.RangeListX.04, PD.Point.RangeListX.05,… and PD.Point.RangeListX.10 if the first 3 point sigma sets were those of common sigma matrix values. These remaining 7 lists give the ranges for sigmas that are exceptions. Exceptions are point sigma values that belong to short isolated sequences of points imbedded in longer sequences of points for which the vast majority of points have a point sigma value equal to a single common value. PD.Point.RangeListX.04(j,k), in this example, is a table that has a row (value of j) for each range of points for which the exceptional value of the point covariance matrix is given by ES.Point.SigmaXYZ(4,k).
To look up the pre-computed error covariance for a point in a lidar data set properly constructed per this CMMD, use the tile number and "ipt" value and query the exception lists, PD.Point.RangeListX.*OrdIndx*(j,k). There will be a list for each value of *OrdIndx*. If the point does fall within an exceptional list, the covariance is looked up in ES.Point.SigmaXYZ(j,k) where j is the integer translated from the string, *OrdIndx*. If the point does not appear in any exception list, proceed to check the common lists, PD.Point.RangeListC.*OrdIndx*(j,k). When the list is found that has a range into which the requested point falls, the common covariance is given by ES.Point.SigmaXYZ(OrdIndx,k), where OrdIndx is the integer corresponding to the evaluated string, *OrdIndx*.

Guidance for classifying an earth frame point covariance as effectively equivalent to a common listed covariance value is laid out in PD.Point.RangeListC.*OrdIndx*(j,k).

Governing quality metrics

See Also

ES.Point.SigmaXYZ(j,k) PD.Point.RangeListC.*OrdIndx*(j,k)
PD.HiIndx.RangeListX.*OrdIndx* PD.TileMap.TileNum(j_tile)

PD.Sort.*

PD.Sort.* data items how the data in the file is sorted.

20.4.80 PD.Sort.NativeOrder [L1,2,3,4,5]

Brief Definition

PD.Sort.NativeOrder indicates whether sorted parameter values increase or decrease in the root data set for the current product's processing path. (The root data set is either an enterprise L1 or the base *processed* data set from which the current data set is derived, as appropriate)

indices	none
Units/representation	Dimensionless Int8
Treatment/usage	When points are originally constructed from the time domain data, they have an order based on the order that fast time vectors were accessed in building each tile. The processor may or may not manage the vectors or tiling in advance of populating the LD.* data in memory. However, when that data is transferred to a file or a communication channel data stream, a deliberate sorting priority and order may be imposed. If it was, PD.Sort.NativeOrder =1 indicates that on a tile basis[179] all parameters in all tiles increase in value or remain unchanged as their position in the tile approaches the last byte in their tile. PD.Sort.NativeOrder = -1 indicates that on a tile basis all parameters in all tiles decrease in value or remain unchanged as their position in the tile approaches the last byte in their tile. PD.Sort.NativeOrder = 2 indicates that on a global basis[180] all parameters in all tiles increase in value or remain unchanged as their position in the tile approaches the last byte in their tile. PD.Sort.NativeOrder = -2 indicates that on a global basis all parameters in all tiles decrease in value or remain unchanged as their position in the tile approaches the last byte in their tile. This ascend or descend flag value is the companion to PD.Sort.NativePri(j), which indicates the priority (first by, then by) in which the parameter sorts were carried out to order the data in the first (that is, "base") stored data blocks to be stored or transmitted for the current data set or its precursors, if there were any.
Governing quality metrics	
See Also	PD.Sort.NativePri(j),

20.4.81 PD.Sort.NativePri(j) [L1,2,3,4,5]

Brief Definition	PD.Sort.NativePri(j) is the global data ordering priority used when the original tiling was imposed on the precursor to the current data set. If this is a level-1 data set, then PD.Sort.NativePri(j) applies to this data set implicitly.
indices	Index, j, has values 1,2,3,4,5,6,7

[179] This means that the sort priority starts over for each tile

[180] This means that the sort priority is global so that tiles are just pieces of a global monotonic sort . Lower number tiles would likely have only one channel if the channel is the priority sort

Units/representation Dimensionless, UInt8

Treatment/usage The global initial tiling data ordering is that which the data had in its original tiling. This could be a carry over of an ordering that existed before assigning the data to tiles at all. The "priority" refers to the precedence in which the parameters were sorted (first by, then by), for example whether points were sorted by X value before Y value or vice versa. The scheme used to document sorting here is the same as is used at the tile level. Refer to PF.Sort.PointPriority.*TileNum*(j).

For each value of j corresponding to these point row items

j= 1--X
2--Y
3--Z
4-- pulse number
5-- channel
6-- intensity
7—return number (not the return code itself)

an integer code is present. The codes are:

0--this item is not sorted
1--This item is first sorting priority
2--This item is first "then by" sorting priority
3--This item is next "then by" sorting priority
etc thru 7[181]

Only the code, "0" may be repeated. Codes greater than 0- must be consecutive values (skipping is not permitted as the code is the precedence, but the order may be scrambled).

For an enterprise level-1 data set, slots for j=1,2,3,6,7 should be set to zero[182], as they apply to points. Then the data for PD.Sort.NativePri(j) might be (0,0,0,2,1,0,0).

For an enterprise level-2 or higher data set, no slots need be set to zero, as the native sort is the first L2 ordering in the processing chain[183] and it applies to both points and clips.

Then the data for PD.Sort.NativePri(j) might be (3,2,0,4,0,0,1). In this example, 4 items are sorted on (4 non zero slots in the array). X is sorted third, after return number and Y. There is no sorting on Z, intensity or

[181] Because, as seen below, there are only 7 items to be sorted. Any realization has the highest code equal to the number of items that are sorted on. The example shows that when 4 items are sorted, the highest code used is 4.

[182] Or equivalently, they must be ignored.

[183] Like the L1 case, the native order implicity applies to the current data set if this is the first L2 in the processing chain.

channel and the last sorting parameter is the pulse number indicated by the value 4 in the j=4 slot of the array.

Governing quality metrics

See Also PF.Sort.PointPriority.*TileNum*(j) ES.Point.RtnsCode.*TileNum*(ipt)

20.4.82 PD.Sort.OrderFlag(j_tile) [L2,3,4,5]

Brief Definition PD.Sort.OrderFlag(j_tile) indicates whether the documented sort describes the data in a tile as physically ordered or that of a descriptive vector that contains a list of row index values that if followed would result in retrieving data in a sorted manner.

indices The index, j_tile, ranges over the tiles and is a 1-1 mapping to the value of *TileNum* . The lowest value of j_tile is 1 and the highest value is PD.TileMap.N_Tiles

Units/representation Dimensionless, UInt8

Treatment/usage This conceptual model supports the option to avoid the run time cost of actually re-storing data into new positions after determining the desired order. This is done by supplying a vector that has the "T_ipt" (that is the row index within a tile) values listed in an order that represents a different sorting of the data from the physical sequence. PD.Sort.OrderFlag(j_tile) is an indicator whether this vector is in use on a tile by tile basis. If this flag is either set to zero for a given tile or not populated at all, the PD.Sort.OrderVec.*TileNum*(j) is not used and the sort description applies to the physically stored data. If this flag is set to 1 for any tile, the sort description data applies to the PD.Sort.OrderVec.*TileNum*(j).

Governing quality metrics Must populate consistently with sorting strategy.

See Also PD.Sort.OrderVec.*TileNum*(j)

20.4.83 PD.Sort.OrderVec.*TileNum*(j) [L2,3,4,5]

Brief Definition PD.Sort.OrderVec.*TileNum*(j) provides an ordered list of tile row numbers corresponding to a documented sort order that differs from the physical storage/stream order.

indices Index, j, ranges from 1 to PD.TileMap.N_Rows(j_tile) that is, from 1 to the number of rows in tile *TileNum*

Units/representation Dimensionless, UInt32

Treatment/usage This conceptual model supports the option to avoid the run time cost of actually re-storing data into new positions after determining the desired order.

This is done by supplying a vector that has the "T_ipt" (that is the row index within a tile) values listed in an order that represents a different sorting of the data from the physical sequence.

The value of index j is the value of the index of the actual row in the tile. The value of PD.Sort.OrderVec.*TileNum*(j) for each j is the new row number. So a tile stored in rows 1,2,3,4 might be resorted to 4,3,2,1 by populating PD.Sort.OrderVec.TileNum(j)= (4,3,2,1).

Governing quality metrics

See Also PD.Sort.OrderFlag(j_tile)

PD.TileMap.*

PD.TileMap.* parameters support the selection of a tile that has geographically desired data.

20.4.84 PD.TileMap.AdjacencyPts(n_tile) [L2,3,4,5]

Brief Definition PD.TileMap.AdjacencyPts(n_tile) gives the list of the horizontal tile locations on a tile grid that have actual point bearing tile data.

indices The index, n_tile, ranges from PD.LoIndx.AdjacentP(1) to PD.HiIndx.AdjacentP(1), that is, over the horizontal tile grid.

Units/representation Dimensionless, UInt16

Treatment/usage This data item is to be populated in the common case where the data is tiled by horizontal geographic position. In this case, the tiles should be sorted on a **fourth quadrant row major** grid. An illustration of a 5row x 6 column horizontal tile map grid is given below. Each cell is filled with the information [(i,j) n_tile] where
"i" is the row number in the tile grid
"j" is the column number in the tile grid
"n_tile" is the linear sequence number of row major "storage" in the tile grid. We use quotes around "storage" to indicate that this is about logical organization only. The actual physical storage is not really governed by this definition. This description is analogous to row major array storage in computer memory.

(1,1) 1	(1,2) 2	(1,3) 3	(1,4) 4	(1,5) 5	(1,6) 6
(2,1) 7	(2,2) 8	(2,3) 9	(2,4) 10	(2,5) 11	(2,6) 12
(3,1) 13	(3,2) 14	(3,3) 15	(3,4) 16	(3,5) 17	(3,1) 18
(4,1) 19	(4,2) 20	(4,3) 21	(4,4) 22	(4,5) 23	(4,1) 24
(5,1) 25	(5,2) 26	(5,3) 27	(5,4) 28	(5,5) 29	(5,6) 30

The total number of rows and columns in the spatial tile map is derived from PD.LoIndx.AdjacentP(j) and PD.HiIndx.AdjacentP(j). Note, that the tile Map grid described in the above figure is not a data item that is ever populated and present in a CMMD compliant data set. It is a figure that can be constructed from PD.LoIndx.AdjacentP(j) and PD.HiIndx.AdjacentP(j) alone.

Now, let us consider the shading pattern in the tile map grid illustrated above. Shaded cells represent the tile map locations with actual populated tiles present in the data set. The unshaded cells represent tile positions on the grid for which there is no lidar data in the current data set. The shaded area is described by PD.TileMap.AdjacentP(n_tile). The index, n_tile, runs over the range of tiles in the grid. The number of distinct "n_tile" values is the number of tile positions in the grid, which is the product of the number of rows and number of columns. So the number of n_tile values is [PD.HiIndx.AdjacentP(1)-PD.LoIndx.AdjacentP(1) +1]. For every value of n_tile, the value of PD.TileMap.AdjacentP(n_tile) is to be populated as follows:

If there is populated tile data in the file for the tile grid position n_tile, then
PD.TileMap.AdjacencyPts(n_tile) = j_tile,

where j_ile is the value of j_tile for the *TileNum* whose data is at the tile grid position, n_tile.

If there is **no** populated tile data in the file for the tile grid position n_tile, then
PD.TileMap.AdjacencyPts(n_tile) = -1.

So, for the 22 tiles in the file that would be described by the above illustration of the tile grid we have:
PD.TileMap.AdjacencyPts(1)= -1, PD.TileMap.AdjacencyPts(2)= 1,
PD.TileMap.AdjacencyPts(3)= 2, PD.TileMap.AdjacencyPts(4)= 3,
PD.TileMap.AdjacencyPts(5)= -1, PD.TileMap.AdjacencyPts(6)= -1,
PD.TileMap.AdjacencyPts(7)=4 , PD.TileMap.AdjacencyPts(8)=5 ,
PD.TileMap.AdjacencyPts(9)= 6, PD.TileMap.AdjacencyPts(10)= 7,
PD.TileMap.AdjacencyPts(11)=8 , PD.TileMap.AdjacencyPts(12)= 9,
PD.TileMap.AdjacencyPts(13)=10 , PD.TileMap.AdjacencyPts(14)=11 ,
PD.TileMap.AdjacencyPts(15)=12 , PD.TileMap.AdjacencyPts(16)=13 ,
PD.TileMap.AdjacencyPts(17)=14 , PD.TileMap.AdjacencyPts(18)= 15,
PD.TileMap.AdjacencyPts(19)= -1, PD.TileMap.AdjacencyPts(20)=16 ,
PD.TileMap.AdjacencyPts(21)=17 , PD.TileMap.AdjacencyPts(22)=18 ,
PD.TileMap.AdjacencyPts(23)=19 , PD.TileMap.AdjacencyPts(24)= -1 ,
PD.TileMap.AdjacencyPts(25)= -1 , PD.TileMap.AdjacencyPts(26)=20 ,

PD.TileMap.AdjacencyPts(27)=21 , PD.TileMap.AdjacencyPts(28)=22 ,
PD.TileMap.AdjacencyPts(29)= -1, PD.TileMap.AdjacencyPts(30)= -1,

Where we have built the case where the first 22 tiles, j_tile=1 through j_tile=22 are point bearing tiles that are positioned on the horizontal grid.

So now we have a solution to the following problem:
We are looking at a point tile with tile number index = **j_tile,** where we use bold letters to indicate a specific value of j_tile, such as 14. Now, provide the value of j_tile for an adjacent tile in a specific direction on the tile grid. Suppose we have selected the value j_tile=11. This is at tile map position (i,j)=(3,2), given the illustration values of PD.TileMap.AdjacencyPts(n_tile) above, and
PD.LoIndx.AdjacentP(1) = 1 PD.HiIndx.AdjacentP(1) = 30
PD.LoIndx.AdjacentP(2) = 1 PD.HiIndx.AdjacentP(2) = 6
PD.LoIndx.AdjacentP(3) = 1
We want to know the j_tile value of the tile that is one row up in the same column, that is at (2,2) on the tile grid. We will also ask for the next diagonal down- right tile, that is at (4,3). The sought after j_tile values are found simply by the following procedure:
Let J=the number of columns in the row major tile grid.

Then J= [PD.HiIndx.AdjacentP(2)- PD.LoIndx.AdjacentP(2) +1]

i.e. J= 6 in the illustration.

And for convenience, I= the number of rows in the row major tile grid

with I= [PD.HiIndx.AdjacentP(1)- PD.LoIndx.AdjacentP(1) +1]/J

i.e. I=5 in the illustration

Given a tile grid location of (i,j), where
PD.LoIndx.AdjacentP(3) ≤ i ≤ PD.LoIndx.AdjacentP(3) + I-1, and
PD.LoIndx.AdjacentP(2) ≤ j ≤ PD.HiIndx.AdjacentP(2)
We can evaluate n_tile for (i,j) according to:

n_tile(i,j) = J×(i- PD.LoIndx.AdjacentP(3)) + (j- PD.LoIndx.AdjacentP(2) +1)

For this illustration n_tile(3,2)= 6×(3-1) +(2-1+1)=14

Then performing the lookup, PD.TileMap.AdjacencyPts(14)=11 would provide the j_tile for the tile at tile grid location (3,2). If we had only been given that we are looking at j_tile=11 as the starting tile, then we would find 11 in the PD.TileMap.AdjacencyPts(n_tile) list at n_tile=14.

Given a value of n_tile on the grid given the ranges of i and j defined in terms of PD.Lo/HiIndx.AdjacentP(j) above,

i(n_tile)= Trunc[184][n_tile/J] + PD.LoIndx.AdjacentP(3), and
j(n_tile)= n_tile –[J×(i- PD.LoIndx.AdjacentP(3))] +
PD.LoIndx.AdjacentP(2)-1
For this illustration, i(14)=Trunc[14/6] +1=3
j(14)=14-6×(3-1) +1-1=2
So, given that we want the value of j_tile for the tile at grid position (2,2) and (4,3) we convert each (i,j) pair to a value of n_tile per the above formulas and would find for our illustration that n_tile(2,2)=8 and
n_tile(4,3)=21. Performing the look-up, j_tile(2,2)= PD.TileMap.AdjacencyPts(8)= 5 and j_tile(4,3)= PD.TileMap.AdjacencyPts(21)= 19. Note that had we asked for j_tile for (i,j)= (1,5) the look up for n_tile=5 would have given j_tile= -1 which would mean that there is no tile to request at that position. Any implementation of tile location based on PD.TileMap.AdjacencyPts(n_tile) and PD.Lo/HiIndx.AdjacentP(j) would need to separately enforce trapping on conditions where (i,j) lie outside the tile grid such that there is no j_tile look up support for the apparently requested value of n_tile.

Governing quality metrics

See Also PD.HiIndx.AdjacentP(j) PD.TileMap.AdjacencyWFs(n_tile)

20.4.85 PD.TileMap.AdjacencyWFs(n_tile) [L1,2,5]

Brief Definition PD.TileMap.AdjacencyWFs(n_tile) gives the list of the horizontal tile locations on a tile grid that have actual waveform bearing tile data.

indices The index, n_tile, ranges from PD.LoIndx.AdjacentW(1) to PD.HiIndx.AdjacentW(1), that is, over the horizontal tile grid.

Units/representation Dimensionless, UInt16

Treatment/usage This data item is to be populated in the common case where the data is tiled by horizontal geographic position. In this case, the tiles should be sorted on a **fourth quadrant row major** grid. An illustration of a 5 row x 6 column horizontal tile map grid is given below. Each cell is filled with the information [(i,j) n_tile] where
"i" is the row number in the tile grid
"j" is the column number in the tile grid
"n_tile" is the linear sequence number of row major "storage" in the tile grid. We use quotes around "storage" to indicate that this is about logical

[184] Round down the ratio

organization only. The actual physical storage is not really governed by this definition. This description is analogous to row major array storage in computer memory.

(1,1) 1	(1,2) 2	(1,3) 3	(1,4) 4	(1,5) 5	(1,6) 6
(2,1) 7	(2,2) 8	(2,3) 9	(2,4) 10	(2,5) 11	(2,6) 12
(3,1) 13	(3,2) 14	(3,3) 15	(3,4) 16	(3,5) 17	(3,1) 18
(4,1) 19	(4,2) 20	(4,3) 21	(4,4) 22	(4,5) 23	(4,1) 24
(5,1) 25	(5,2) 26	(5,3) 27	(5,4) 28	(5,5) 29	(5,6) 30

The total number of rows and columns in the spatial tile map is derived from PD.LoIndx.AdjacentW(j) and PD.HiIndx.AdjacentW(j). Note, that the tile Map grid described in the above figure is not a data item that is ever populated and present in a CMMD compliant data set. It is a figure that can be constructed from PD.LoIndx.AdjacentW(j) and PD.HiIndx.AdjacentW(j) alone.

Now, let us consider the shading pattern in the tile map grid illustrated above. Shaded cells represent the tile map locations with actual populated tiles present in the data set. The unshaded cells represent tile positions on the grid for which there is no lidar data in the current data set. The shaded area is described by PD.TileMap.AdjacencyWFs(n_tile). The index, n_tile, runs over the range of tiles in the grid. The number of distinct "n_tile" values is the number of tile positions in the grid, which is the product of the number of rows and the number of columns. So the number of n_tile values is [PD.HiIndx.AdjacentW(1)-PD.LoIndx.AdjacentW(1) +1]. For every value of n_tile, the value of PD.TileMap.AdjacencyWFs(n_tile) is to be populated as follows:

> If there is populated tile data in the file for the tile grid position n_tile, then
> PD.TileMap.AdjacencyWFs(n_tile) = j_tile,
>> where j_tile is the value of j_tile for the *TileNum* whose data is at the tile grid position, n_tile.
>
> If there is **no** populated tile data in the file for the tile grid position n_tile, then
> PD.TileMap.AdjacencyWFs(n_tile) = -1.

So, for the 22 tiles in the file that would be described by the above illustration of the tile grid we have:
PD.TileMap.AdjacencyWFs(1)= -1, PD.TileMap.AdjacencyWFs(2)= 23,
PD.TileMap.AdjacencyWFs(3)= 24, PD.TileMap.AdjacencyWFs(4)= 25,

PD.TileMap.AdjacencyWFs(5)= -1, PD.TileMap.AdjacencyWFs(6)= -1,
PD.TileMap.AdjacencyWFs(7)=26 , PD.TileMap.AdjacencyWFs(8)=27 ,
PD.TileMap.AdjacencyWFs(9)= 28, PD.TileMap.AdjacencyWFs(10)= 29,
PD.TileMap.AdjacencyWFs(11)=30 , PD.TileMap.AdjacencyWFs(12)= 31,
PD.TileMap.AdjacencyWFs(13)=32 , PD.TileMap.AdjacencyWFs(14)=33 ,
PD.TileMap.AdjacencyWFs(15)=34 , PD.TileMap.AdjacencyWFs(16)=35 ,
PD.TileMap.AdjacencyWFs(17)=36 , PD.TileMap.AdjacencyWFs(18)= 37,
PD.TileMap.AdjacencyWFs(19)= -1, PD.TileMap.AdjacencyWFs(20)=38 ,
PD.TileMap.AdjacencyWFs(21)=39 , PD.TileMap.AdjacencyWFs(22)=40 ,
PD.TileMap.AdjacencyWFs(23)=41 , PD.TileMap.AdjacencyWFs(24)= -1 ,
PD.TileMap.AdjacencyWFs(25)= -1 , PD.TileMap.AdjacencyWFs(26)=42 ,
PD.TileMap.AdjacencyWFs(27)=43 , PD.TileMap.AdjacencyWFs(28)=44 ,
PD.TileMap.AdjacencyWFs(29)= -1, PD.TileMap.AdjacencyWFs(30)= -1,

Where we have built the case where the second 22 tiles, j_tile=23 through j_tile=44 are waveform bearing tiles that are positioned on the horizontal grid.

So now we have a solution to the following problem:
We are looking at a point tile with tile number index = **j_tile,** where we use bold letters to indicate a specific value of j_tile, such as 36. Now, provide the value of j_tile for an adjacent tile in a specific direction on the tile grid. Suppose we have selected the value j_tile=33. This is at tile map position (i,j)=(3,2), given the illustration values of PD.TileMap.AdjacencyWFs(n_tile) above, and
PD.LoIndx.AdjacentW(1) = 1 PD.HiIndx.AdjacentW(1) = 30
PD.LoIndx.AdjacentW(2) = 1 PD.HiIndx.AdjacentW(2) = 6
PD.LoIndx.AdjacentW(3) = 1
We want to know the j_tile value of the tile that is one row up in the same column, that is at (2,2) on the tile grid. We will also ask for the next diagonal down- right tile, that is at (4,3). The sought after j_tile values are found simply by the following procedure:
Let J=the number of columns in the row major tile grid.
Then J= [PD.HiIndx.AdjacentW(2)- PD.LoIndx.AdjacentW(2) +1]
i.e. J= 6 in the illustration.
And for convenience, I= the number of rows in the row major tile grid
with I= [PD.HiIndx.AdjacentW(1)- PD.LoIndx.AdjacentW(1) +1]/J
i.e. I=5 in the illustration
Given a tile grid location of (i,j), where
PD.LoIndx.AdjacentW(3) $\leq$ i $\leq$ PD.LoIndx.AdjacentW(3) + I-1, and
PD.LoIndx.AdjacentW(2) $\leq$ j $\leq$ PD.HiIndx.AdjacentW(2)
We can evaluate n_tile for (i,j) according to:

n_tile(i,j) = J×(i- PD.LoIndx.AdjacentW(3)) + (j- PD.LoIndx.AdjacentW(2) +1)
For this illustration n_tile(3,2)= 6×(3-1) +(2-1+1)=14
Then performing the lookup, PD.TileMap.AdjacencyWFs(14)=33 would provide the j_tile for the tile at tile grid location (3,2). If we had only been given that we are looking at j_tile=33 as the starting tile, then we would find 33 in the PD.TileMap.AdjacencyWFs(n_tile) list at n_tile=14.

Given a value of n_tile on the grid given the ranges of i and j defined in terms of PD.Lo/HiIndx.AdjacentW(j) above,
i(n_tile)= Trunc[184][n_tile/J] + PD.LoIndx.AdjacentW(3), and
j(n_tile)= n_tile –[J×(i- PD.LoIndx.AdjacentW(3))] + PD.LoIndx.AdjacentW(2)-1
For this illustration, i(14)=Trunc[14/6] +1=3
j(14)=14-6×(3-1) +1-1=2
So, given that we want the value of j_tile for the tile at grid position (2,2) and (4,3) we convert each (i,j) pair to a value of n_tile per the above formulas and would find for our illustration that n_tile(2,2)=8 and
n_tile(4,3)=21. Performing the look-up, j_tile(2,2)= PD.TileMap.AdjacencyWFs(8)= 27 and j_tile(4,3)= PD.TileMap.AdjacencyWFs(21)= 41. Note that had we asked for j_tile for (i,j)= (1,5) the look up for n_tile=5 would have given j_tile= -1 which would mean that there is no tile to request at that position. Any implementation of tile location based on PD.TileMap.AdjacencyWFs(n_tile) and PD.Lo/HiIndx.AdjacentW(j) would need to separately enforce trapping on conditions where (i,j) lie outside the tile grid such that there is no j_tile look up support for the apparently requested value of n_tile.

Governing quality metrics

See Also PD.HiIndx.AdjacentW(j) PD.TileMap.AdjacencyPts(n_tile)

20.4.86 PD.TileMap.ChanList(j_tile,k) [L1,2,3,4,5]

Brief Definition PD.TileMap.ChanList(j_tile,k) gives the list of the occurring channels for each tile.

indices The index, j_tile, ranges over the tiles and is a 1-1 mapping to the value of *TileNum* . The lowest value of j_tile is 1 and the highest value is PD.TileMap.N_Tiles

	Index, k ranges from PD.LoIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m), that is, over the channels of the current data set.
Units/representation	Dimensionless, UInt8
Treatment/usage	For a lidar that has only one channel, all data will be from that single channel and the choice of which channels might be represented in any tile does not exist. In such a case, PD.TileMap.ChanList(j_tile,k) need not be populated. However, when more than one channel descriptor set is provided (as could be the case for a lidar working in multiple wavelengths), PD.TileMap.ChanList(j_tile,k) gives the list of channel presence (=1 if a channel's data is included in a given tile, =0 if that channel is not present in the tile) for each tile as indexed by j_tile. PD.TileMap.ChanList(j_tile, 'LoIndx') has the indicator for the lowest channel number index with data present in the data set. PD.TileMap.ChanList(j_tile,'HiIndx') has the indicator for the highest channel number index with data present in the data set. When there is a choice of channels, but the tiles are constrained to a single channel for any tile, PD.TileMap.DedictdTile.Cnstrnt*n*.Name should be used in lieu of PD.TileMap.ChanList(j_tile,k).
Governing quality metrics	
See Also	PD.HiIndx.Channlzn.ChRg(m) PD.HiIndx.Channlzn.Rg_m PD.TileMap.DedictdTile.Cnstrnt1.Name

20.4.87 PD.TileMap.DedictdTile.Cnstrnt1.Name [L1,2,3,4,5]

Brief Definition	PD.TileMap.DedictdTile.Cnstrnt1.Name gives the identifying code of the first parameter that is a single value for every tile when a "dedicated tile" strategy is used.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.TileMap.DedictdTile.Cnstrnt1.Name will only be populated when an explicit dedicated tile strategy is employed. An explicit dedicated tile strategy is when there is a range of values available for one of the listed items coded below and when the tiles are populated such that in any single tile, only a single value of the constrained parameter is present. PD.TileMap.DedictdTile.Cnstrnt1.Name identifies the first parameter that is a single value for every tile.

Codes are assigned per the following table:

Value of * Cnstrnt1.Name	Constrained quantity
1	Channel#
2	Return# (not the return code)
3	Parent# (Level-4 only)
4	Scan Phase
5…	*Table may be extended in later releases to accommodate emerging needs*

See PD.TileMap.DedictdTile.Cnstrnt1.Val(j) for further explanation of quantities subject to single value constraint in a tile.
See dedicated tile table illustration in 10.2.1.5.
PD.TileMap.DedictdTile.Cnstrnt1.Name is the header for data represented by PD.TileMap.DedictdTile.Cnstrnt1.Val(j) in the first column of figure 10.2.1-7.

Governing quality metrics

See Also

PD.TileMap.DedictdTile.Cnstrnt1.Val(j)
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j)

20.4.88 PD.TileMap.DedictdTile.Cnstrnt1.Val(j) [L1,2,3,4,5]

Brief Definition

PD.TileMap.DedictdTile.Cnstrnt1.Val(j) gives the value of the first constrained data item for the jth tile group.

indices

Index, j, ranges from PD.LoIndx.TileMap.DedictdTile to PD.HiIndx.TileMap.DedictdTile, that is, over the dedicated tile groups

Units/representation

Dimensionless, UInt8

Treatment/usage

The dedicated tile strategy is explained in 10.2.1.5. Some of the elements of a "point row" have only a few possible values and data is also naturally of particular interest when sorted by these values. For example, it may be much more convenient in multiple wavelength systems to not have the points intermingled by wavelength, but rather presented in separate tiles.

PD.TileMap.DedictdTile.Cnstrnt1.Val(j) should only be populated when a dedicated tile strategy is used for a given data set and will be present in conjunction with PD.TileMap.DedictdTile.Cnstrnt1.Name, PD.TileMap.DedictdTile.TileNumValLow(j), and PD.TileMap.DedictdTile.TileNumValHi(j).

A "tile group" corresponds to a row in figure 10.2.1-7. A row in that figure is characterized by a unique combination of the values of the constraint

parameters. If there is only one parameter being used as a dedicated constraint, then there is a distinct value of the constraint parameter for each tile group, as there is only one element in the combination. In the illustrated case, two constraints are used, resulting in repetitions of single element values in different combinations.

To illustrate the use of PD.TileMap.DedictdTile.Cnstrnt1.Val(j), the channel number [determines the appropriate value of the index in SC.Array.FilterCode(ia)] might be one of the items used in a dedicated tile strategy. In this case, channel number might be a code used in PD.TileMap.DedictdTile.Cnstrnt1.Name.
PD.TileMap.DedictdTile.Cnstrnt1.Val(j) gives the value of the first dedicated tile parameter for the dedicated tile group indicated by the value of j.
So, if there are two channels as defined in PF.Channlzn.Descr(j)/ PD.HiIndx.Channlzn.ChRg(m), and the channel number is used as the first dedicated tile constraint parameter, a tile dedicated to the second channel would lie in a tile group (that is, a choice of j) where PD.TileMap.DedictdTile.Cnstrnt1.Val(j) =2.
PD.TileMap.DedictdTile.Cnstrnt1.Val(j) are the data represented by the first column of figure 10.2.1-7.

If a constraint does not apply to all tile groups, when describing a dedicated tile group for which it does not apply, one must insert a value that codes for N/A. The N/A codes to be inserted into PD.TileMap.DedictdTile.Cnstrnt*N*.Val(j), are given below:

constraint	**N/A value**
1 Channel#	0
2 Return# (not the return code)	0
3 Parent# (Level-4 only)	0
4 Scan Phase	-10.00E00

This table covers the most likely dedicated constraint items. As new dedicated constraint N/A codes become necessary, this paragraph should be amended by extending this table. It must be kept in synch with the name code tables presented, for example, in PD.TileMap.DedictdTile.Cnstrnt1.Name.

Governing quality metrics

See Also

PD.HiIndx.TileMap.DedictdTile PD.TileMap.DedictdTile.Cnstrnt1.Name
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j) PF.Channlzn.Descr(j)

20.4.89 PD.TileMap.DedictdTile.Cnstrnt2.Name [L1,2,3,4,5]

Brief Definition: PD.TileMap.DedictdTile.Cnstrnt2.Name gives the identifying code of the second parameter that is a single value for every tile when a "dedicated tile" strategy is used.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.TileMap.DedictdTile.Cnstrnt2.Name will only be populated when an explicit dedicated tile strategy is employed. An explicit dedicated tile strategy is when there is a range of values available for one of the listed items coded below and when the tiles are populated such that in any single tile, only a single value of the constrained parameter is present.
PD.TileMap.DedictdTile.Cnstrnt2.Name identifies the second parameter that is a single value for for every tile
Codes are assigned per the following table:

Value of * Cnstrnt2.Name	Constrained quantity
1	Channel#
2	Return# (not the return code)
3	Parent# (Level-4 only)
4	Scan Phase
5…	*Table may be extended in later releases to accommodate emerging needs*

See PD.TileMap.DedictdTile.Cnstrnt2.Val(j) for further explanation of quantities subject to single value constraint in a tile.
See dedicated tile table illustration in 10.2.1.5.
PD.TileMap.DedictdTile.Cnstrnt2.Name is the header for data represented in the second column of figure 10.2.1-7. [Note that the illustration in 10.2.1-7 uses the channel as the first dedicated tile constraint. Data set writers get to determine the choice of dedicated tile constraints by choosing the values of constraint names.]

Governing quality metrics

See Also: PD.TileMap.DedictdTile.Cnstrnt2.Val(j)
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j)

20.4.90 PD.TileMap.DedictdTile.Cnstrnt2.Val(j) [L1,2,3,4,5]

Brief Definition: PD.TileMap.DedictdTile.Cnstrnt2.Val(j) gives the value of the second constrained data item for the jth tile group.

indices	Index, j, ranges from PD.LoIndx.TileMap.DedictdTile to PD.HiIndx.TileMap.DedictdTile, that is over the dedicated tile groups
Units/representation	Dimensionless, UInt8
Treatment/usage	The dedicated tile strategy is explained in 10.2.1.5. Some of the elements of a "point row" have only a few possible values and data is also naturally of particular interest when sorted by these values. For example, it may be much more convenient in multiple wavelength systems to not have the points intermingled by wavelength, but rather presented in separate tiles. PD.TileMap.DedictdTile.Cnstrnt2.Val(j) should only be populated when a dedicated tile strategy is used for a given data set and will be present in conjunction with PD.TileMap.DedictdTile.Cnstrnt2.Name, PD.TileMap.DedictdTile.TileNumValLow(j), and PD.TileMap.DedictdTile.TileNumValHi(j). A "tile group" corresponds to a row in figure 10.2.1-7. A row in that figure is characterized by a unique combination of values for the values of the constraint parameters. If there is only one parameter being used as a dedicated constraint, then there is a distinct value of the constraint parameter for each tile group, as there is only one element in the combination. In the illustrated case, two constraints are used, resulting in repetitions of single element values in different combinations. To illustrate the use of PD.TileMap.DedictdTile.Cnstrnt2.Val(j), the channel number [determines the appropriate value of the index in SC.Array.FilterCode(ia)] might be one of the items used in a dedicated tile strategy. In this case, channel number might be a code used in PD.TileMap.DedictdTile.Cnstrnt2.Name. PD.TileMap.DedictdTile.Cnstrnt2.Val(j) gives the value of the second dedicated tile parameter for the dedicated tile group indicated by the value of j. So, if there are two channels as defined in PF.Channlzn.Descr(j)/ PD.HiIndx.Channlzn.ChRg(m), and the channel number is used as the second dedicated tile constraint parameter, a tile dedicated to the second channel would lie in a tile group where PD.TileMap.DedictdTile.Cnstrnt2.Val(j) =2. PD.TileMap.DedictdTile.Cnstrnt2.Val(j) is the data represented by the second column of figure 10.2.1-7. If a constraint does not apply to all tile groups, when describing a dedicated tile group for which it does not apply, one must insert a value that codes for N/A. The N/A codes to be inserted into

PD.TileMap.DedictdTile.Cnstrnt*N*.Val(j), are given below:

constraint	N/A value
1 Channel#	0
2 Return# (not the return code)	0
3 Parent# (Level-4 only)	0
4 Scan Phase	-10.00E00

This table covers the most likely dedicated constraint items. As new dedicated constraint N/A codes become necessary, this paragraph should be amended by extending this table. It must be kept in synch with the name code tables presented, for example, in PD.TileMap.DedictdTile.Cnstrnt2.Name.

Governing quality metrics

See Also

PD.HiIndx.TileMap.DedictdTile PD.TileMap.DedictdTile.Cnstrnt2.Name
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j)

20.4.91 PD.TileMap.DedictdTile.Cnstrnt3.Name [L1,2,3,4,5]

Brief Definition

PD.TileMap.DedictdTile.Cnstrnt3.Name gives the identifying code of the third parameter that is a single value for every tile when a “dedicated tile” strategy is used.

indices

none

Units/representation

Dimensionless, UInt8

Treatment/usage

PD.TileMap.DedictdTile.Cnstrnt3.Name will only be populated when an explicit dedicated tile strategy is employed for three items. An explicit dedicated tile strategy is when there is a range of values available for one of the listed items coded below and when the tiles are populated such that in any single tile, only a single value of the constrained parameter is present. PD.TileMap.DedictdTile.Cnstrnt3.Name identifies the third parameter that is a single value for for every tile

Codes are assigned per the following table:

Value of * Cnstrnt3.Name	Constrained quantity
1	Channel#
2	Return# (not the return code)
3	Parent# (Level-4 only)
4	Scan Phase
5…	*Table may be extended in later releases to accommodate emerging needs*

See PD.TileMap.DedictdTile.Cnstrnt3.Val(j) for further explanation of quantities subject to single value constraint in a tile.

See dedicated tile table illustration in 10.2.1.5.
PD.TileMap.DedictdTile.Cnstrnt3.Name would be the header for data represented in the third column of figure 10.2.1-7 if the table were extended to 3 constraints.

Governing quality metrics

See Also

PD.TileMap.DedictdTile.Cnstrnt3.Val(j)
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j)

20.4.92 PD.TileMap.DedictdTile.Cnstrnt3.Val(j) [L1,2,3,4,5]

Brief Definition

PD.TileMap.DedictdTile.Cnstrnt3.Val(j) gives the value of the third constrained data item for the jth tile group.

indices

Index, j, ranges from PD.LoIndx.TileMap.DedictdTile to PD.HiIndx.TileMap.DedictdTile, that is over the dedicated tile groups

Units/representation

Dimensionless, UInt8

Treatment/usage

The dedicated tile strategy is explained in 10.2.1.5. Some of the elements of a "point row" have only a few possible values and data is also naturally of particular interest when sorted by these values. For example, it may be much more convenient in multiple wavelength systems to not have the points intermingled by wavelength, but rather presented in separate tiles.

PD.TileMap.DedictdTile.Cnstrnt3.Val(j) should only be populated when a dedicated tile strategy is used for 3 constraints in a given data set and will be present in conjunction with PD.TileMap.DedictdTile.Cnstrnt3.Name, PD.TileMap.DedictdTile.TileNumValLow(j), and PD.TileMap.DedictdTile.TileNumValHi(j).

A "tile group" corresponds to a row in figure 10.2.1-7. A row in that figure is characterized by a unique combination of values for the values of the constraint parameters. If there is only one parameter being used as a dedicated constraint, then there is a distinct value of the constraint parameter for each tile group, as there is only one element in the combination. In the illustrated case, two constraints are used, resulting in repetitions of single element values in different combinations.

The use of PD.TileMap.DedictdTile.Cnstrnt3.Val(j), is analogous to the explanations in PD.TileMap.DedictdTile.Cnstrnt1.Val(j) and PD.TileMap.DedictdTile.Cnstrnt2.Val(j).
PD.TileMap.DedictdTile.Cnstrnt3.Val(j) would be the data represented by the

third column of figure 10.2.1-7 if it were extended to illustrate three constraints.

If a constraint does not apply to all tile groups, when describing a dedicated tile group for which it does not apply, one must insert a value that codes for N/A. The N/A codes to be inserted into PD.TileMap.DedictdTile.Cnstrnt*N*.Val(j), are given below:

constraint	**N/A value**
1 Channel#	0
2 Return# (not the return code)	0
3 Parent# (Level-4 only)	0
4 Scan Phase	-10.00E00

This table covers the most likely dedicated constraint items. As new dedicated constraint N/A codes become necessary, this paragraph should be amended by extending this table. It must be kept in synch with the name code tables presented, for example, in PD.TileMap.DedictdTile.Cnstrnt3.Name.

Governing quality metrics

See Also

PD.HiIndx.TileMap.DedictdTile PD.TileMap.DedictdTile.Cnstrnt3.Name
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.TileNumValHi(j)

20.4.93 PD.TileMap.DedictdTile.TileNumValHi(j) [L1,2,3,4,5]

Brief Definition

PD.TileMap.DedictdTile.TileNumValHi(j) is an integer corresponding to the highest sorting value (i.e., last) of *TileNum* in dedicated tile group(j).

indices

Index, j, ranges from PD.LoIndx.TileMap.DedictdTile to PD.HiIndx.TileMap.DedictdTile, that is over the dedicated tile groups

Units/representation

Dimensionless, UInt16

Treatment/usage

The dedicated tile strategy is explained in 10.2.1.5. Some of the elements of a "point row" have only a few possible values and data is also naturally of particular interest when sorted by these values. For example, it may be much more convenient in multiple wavelength systems to not have the points intermingled by wavelength, but rather presented in separate tiles.

A "tile group", consists of a consecutive run of tiles as indexed by j_tile and corresponds to a row in figure 10.2.1-7. A row in that figure is characterized by a unique combination of values for the constraint parameters. If there is only one parameter being used as a dedicated constraint, then there is a distinct value of the constraint parameter for each tile group, as there is only

one element in the combination. In the illustrated case, two constraints are used, resulting in repetitions of single element values in different combinations.

In the table of figure 10.2.1-7, the consecutive tiles of a tile group are indicated in the right two columns. The right most column is formed from PD.TileMap.DedictdTile.TileNumValHi(j) which represents the highest sorting value (i.e. last) of j_tile, the integer that is mapped to *TileNum* by PD.TileMap.TileNum(j_tile), in dedicated tile group(j). PD.TileMap.DedictdTile.TileNumValHi(j) should only be populated when an explicit dedicated tile strategy is used for a given data set

Governing quality metrics

See Also

PD.HiIndx.TileMap.DedictdTile PD.TileMap.DedictdTile.Cnstrnt1.Name
PD.TileMap.DedictdTile.TileNumValLow(j)
PD.TileMap.DedictdTile.Cnstrnt1.Val(j)

20.4.94 PD.TileMap.DedictdTile.TileNumValLow(j) [L1,2,3,4,5]

Brief Definition

PD.TileMap.DedictdTile.TileNumValLow(j) is an integer corresponding to the lowest sorting value (i.e. first) of *TileNum* in dedicated tile group(j).

indices

Index, j, ranges from PD.LoIndx.TileMap.DedictdTile to PD.HiIndx.TileMap.DedictdTile, that is over the dedicated tile groups

Units/representation

Dimensionless, UInt16

Treatment/usage

The dedicated tile strategy is explained in 10.2.1.5. Some of the elements of a "point row" have only a few possible values and data is also naturally of particular interest when sorted by these values. For example, it may be much more convenient in multiple wavelength systems to not have the points intermingled by wavelength, but rather presented in separate tiles.

A "tile group" corresponds to a row in figure 10.2.1-7. A row in that figure is characterized by a unique combination of values of the constraint parameters. If there is only one parameter being used as a dedicated constraint, then there is a distinct value of the constraint parameter for each tile group, as there is only one element in the combination. In the illustrated case, two constraints are used, resulting in repetitions of single element values in different combinations.

In the table of figure 10.2.1-7, the consecutive tiles of a tile group are indicated in the right two columns. The next-to right most column is formed from PD.TileMap.DedictdTile.TileNumValLow(j) which represents the

lowest sorting value (i.e. first) of j_tile, the integer that is mapped to *TileNum* by PD.TileMap.TileNum(j_tile) in dedicated tile group(j). PD.TileMap.DedictdTile.TileNumValLow(j) should only be populated when an explicit dedicated tile strategy is used for a given data set

Governing quality metrics

See Also PD.TileMap.DedictdTile.Cnstrnt1.Name
PD.TileMap.DedictdTile.TileNumValHi(j)
PD.TileMap.DedictdTile.Cnstrnt1.Val(j)

20.4.95 PD.TileMap.GlblPtTyping [L2,3,4,5]

Brief Definition PD.TileMap.GlblPtTyping encodes the mix of lidar points and synthetic lidar points in the data set.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage Lidar processing from time domain data generates raw lidar points when the directions and round trip delays are transformed into angle-angle-range and finally XYZ point coordinates. These lidar points may be subject to a form of processing that creates new "synthetic" points derived from multiple lidar points each. The synthetic points may replace some or all of the lidar points in a data set. To indicate whether there are any synthetic points in a data set PD.TileMap.GlblPtTyping has the following three code values:
--1 for data set contains only lidar points,
--2 for data set contains only synthetic lidar points,
--3 for data set has mixed points and we need to populate PD.TileMap.PtType.*TileNum*
Note that if the data set has both lidar and synthetic lidar points, they must be of a single type in a single tile.

Governing quality metrics

See Also PD.TileMap.PtType(j_tile)

20.4.96 PD.TileMap.Lat(j_tile,k) [L1, 2,3,4,5]

Brief Definition PD.TileMap.Lat(j_tile,k) gives the latitude of the northernmost and southernmost data in the tile.

indices The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles
Index, k, =1,2

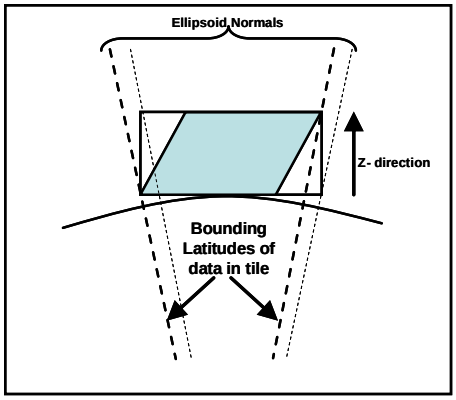
Units/representation Decimal degrees, Float

Treatment/usage PD.TileMap.Lat(j_tile,k) has been provided in order to support documenting

the horizontal coverage of a tile of lidar data. Supplying the latitude extent of a tile's content is a convenient way of documenting tile coverage. The meaning of PD.TileMap.Lat(j_tile,k) is slightly contextual, depending on the enterprise level of the product that it is populated within.

- For enterprise level-1 data, which is time domain data, nominal tile ground coverage computed before the data is processed to XYZ in the earth frame is provided by PD.TileMap.Lat(j_tile,k). This is discussed in 10.1.2.5. [Angle-angle-range data in the sensor frame is a possible level-2 option that would use data computed at the L-1 fidelity level to document tile coverage as well.]
- For enterprise level-2 and above, points in the earth frame have been computed for waveform clips and, commonly, the points are stored in the earth frame. Here, the actual bounding points are to be selected in populating the latitude coverage.

The Cartesian frames span rectangular prism blocks of 3-D space. Owing to the possibility that the data was collected from an off-nadir aspect, the rectangular prism may not be populated with data points uniformly. Furthermore, the z-direction along the bound of the prism may not correspond to the ellipsoid normal passing through the base or top levels. Thus, labeling the tile by the extent of the Cartesian frame could misrepresent the data coverage. PD.TileMap.Lat(j_tile,1) is the latitude of the northernmost point in the tile, where the latitude is determined by dropping an ellipsoid normal through the point to the WGS-84 ellipsoid.



PD.TileMap.Lat(j_tile,2) is the latitude of the southernmost point in the tile, where the latitude is determined by dropping an ellipsoid normal through the point to the WGS-84 ellipsoid.

Governing quality metrics

For small tiles, the ellipsoid normal and z may be close to parallel. It is somewhat simpler to find the latitude via transformation of the X coordinate of the data point, but this introduces a calculation error that will, in general,

be subject to an implementation accuracy requirement. A delta latitude value can represent the errors due to such approximations.[185]

See Also

20.4.97 PD.TileMap.Lon(j_tile,k) [L1,2,3,4,5]

Brief Definition: PD.TileMap.Lon(j_tile,k) gives the longitude of the westernmost and easternmost data in the tile

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles. Index, k, =1,2

Units/representation: Decimal degrees, Float

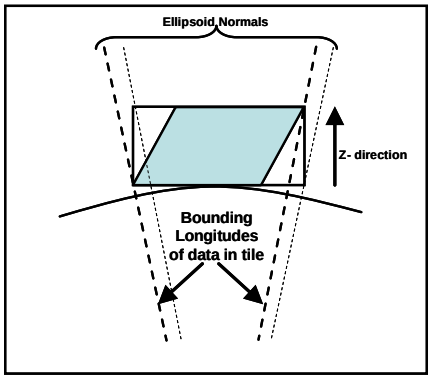
Treatment/usage: PD.TileMap.Lon(j_tile,k) has been provided in order to support documenting the horizontal coverage of a tile of lidar data. Supplying the longitude extent of a tile's content is a convenient way of documenting tile coverage. The meaning of PD.TileMap.Lon(j_tile,k) is slightly contextual, depending on the enterprise level of the product that it is populated within.

- For enterprise level-1 data, which is time domain data, nominal tile ground coverage computed before the data is processed to XYZ in the earth frame is provided by PD.TileMap.Lon(j_tile,k). This is discussed in 10.1.2.5. [Angle-angle-range data in the sensor frame is a possible level-2 option that would use data computed at the L-1 fidelity level to document tile coverage as well.]
- For enterprise level-2 and above, points in the earth frame have been computed for waveform clips and, commonly, the points are stored in the earth frame. Here, the actual bounding points are to be selected in populating the latitude coverage.

The Cartesian frames span rectangular prism blocks of 3-D space. Owing to the possibility that the data was collected from an off-nadir aspect, the rectangular prism may not be populated with data points uniformly. Furthermore, the z-direction along the bound of the prism may not correspond to the ellipsoid normal passing through the base or top levels. Thus, labeling the tile by the extent of the Cartesian frame could misrepresent the data

[185] Since the bounds are not exploitation *results*, the error in the tile bounding latitude is not defined as a metadata item. It is recommended that the implementation spec for this data item demand errors to be small (1 % or less) compared to the latitude extents of data tiles.

coverage. PD.TileMap.Lon(j_tile,1) is the longitude of the westernmost point in the tile, where the longitude is determined by dropping an ellipsoid normal through the point to the WGS-84 ellipsoid.



PD.TileMap.Lon(j_tile,2) is the longitude of the easternmost point in the tile, where the longitude is determined by dropping an ellipsoid normal through the point to the WGS-84 ellipsoid.

Governing quality metrics

For small tiles, the ellipsoid normal and z may be close to parallel. It is somewhat simpler to find the longitude via the translation of the Y coordinate of the data point, but this introduces a calculation error that will, in general, be subject to an implementation accuracy requirement. A delta longitude value can represent the errors due to such approximations[185]

See Also

20.4.98 PD.TileMap.MaxNsamps.*TileNum* [L1,2]

Brief Definition

PD.TileMap.MaxNsamps.*TileNum* gives the number of samples in the longest waveform vector of tile *TileNum*.

indices

None, however, *TileNum*, though part of a tag name, acts somewhat as an index. There are as many instantiations of PD.TileMap.MaxNsamps.*TileNum* as there are tiles of waveform data. The set of instantiated values of *TileNum* provide data in similarly named scalar data items as if we had a vector, PD.TileMap.MaxNsamps(j), where j ranged over the wave form data tiles.

Units/representation

Dimensionless, UInt32

Treatment/usage

Waveform data is stored in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it). LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) is an array sized for the data associated with the waveform bearing tile indicated by *TileNum*. Thus we need to know the length of the longest waveform clip in each waveform bearing tile. PD.TileMap.MaxNsamps.*TileNum* fills that need by providing

the number of samples in the longest waveform vector of PD.WFClp.RxDatVec.*TileNum*(ip,iray,it).
The rule for assigning a maximum waveform vector length for a tile (thereby establishing the range of the third index in the LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) array for that *TileNum* is the following:

> The value of PD.TileMap.MaxNsamps.*TileNum* will apply for all tiles starting with the tile associated with the *TileNum* in PD.TileMap.MaxNsamps.*TileNum* and continuing in ascending order until overridden by a *TileNum* associated with the next highest *TileNum* in sorting order that appears in another PD.TileMap.MaxNsamps.*TileNum*. This would permit not only several maximum waveform vector lengths for a very large number of tiles, but all waveform tiles may have the same maximum waveform length and that can be supplied through just one populated PD.TileMap.MaxNsamps.*TileNum*.

PD.TileMap.MaxNsamps.*TileNum* only applies to tiles bearing waveforms and will not be present in a points only data set.

Governing quality metrics

See Also: PD.WFClp.Nsamps.*TileNum*(ip,iray,j) SC.WF.Rx.Nsamps(ich) SC.WF.TxProfile.Nsamps(ia) LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)

20.4.99 PD.TileMap.N_Rows(j_tile) [L2,3,4,5]

Brief Definition: PD.TileMap.N_Rows(j_tile) is the number of logical rows of data in tiles indexed by j_tile.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles, that is, over all of the tiles.

Units/representation: Dimensionless, UInt32

Treatment/usage: In the vertically tiled data strategy used in this CMMD, we view data vectors as column vectors, each element of which belongs in a single row of a logical data table. See, for example, figure 10.2.1-3 and 10.2.1-4. The elements of a single row of data in the logical data table representing the lidar data all line up in the table, so every data item with the same value of *TileNum* in its tag has the same number of logical rows. That number is given for each tile by PD.TileMap.N_Rows(j_tile). This j_tile may[186] range over both point and

[186] While this data item is essential for point data tiles, there are other mechanisms that are used for sizing pulse and waveform oriented data arrays.

waveform bearing tiles. The number of logical rows is the same as the number of actual rows when a PD.IntTileMap.*.Flg is not populated for a tiled item. When the redundancy reduction scheme of 10.2.1.5 is used, PD.IntTileMap.*.Flg is populated with the actual number of rows in the tile, which is less than the logical number of rows in these cases.
Note that some tiles are point tiles, with this document generally using "ipt" as a symbol for the row index. Other tiles may be pulse tiles, with this document generally using "ip" ("ipi" in some enterprise L2 data items) as a symbol for the row index. A pulse tile and a point tile will not have the same name subfields, not considering *TileNum*. As it is part of a tag for a data item, *TileNum* values must support unique instantiations of vertical tile vectors. See the level-2 specific discussion regarding *ChNum* and *TileNum* in 10.1.2.4. Also, per 10.1.2.4, the mapping of *TileNum* values to j_tile values is one to one.

Governing quality metrics: Correct value must be supplied

See Also: SC.Pulse.SensorPos.*TileNum*(ip,j) PD.TileMap.TileNum(j_tile)

20.4.100 PD.TileMap.N_Tiles [L1,2,3,4,5]

Brief Definition: PD.TileMap.N_Tiles provides the total number of tiles in the data set. This is the top value of j_tile, which ranges over all of the tile identifiers.

indices: none

Units/representation: Dimensionless, UInt16

Treatment/usage: PD.TileMap.N_Tiles is the sum over all point, waveform and pulse tiles (unique values of *TileNum* for points plus unique values of *TileNum* for pulses plus unique values of *TileNum* for waveforms).

Important referencing convention for tile counting:

There are two contexts of the term "tile" or "unique tile" that are used in this CMMD. With respect to the identification of a value of j_tile that corresponds to *TileNum*, the metadata tag sub-field tile designator, we are referring to tiles as blocks of full rows of the logical (database-like) table of data in a CMMD compliant data set. *TileNum* and j_tile identify one of many blocks of rows (vertical extents within the overall data table) in a data set. Thus, there are as many tiles in this context as there are values of j_tile. In this sense, LD.Point.Intensity.*TileNum*(ipt,j), LD.Point.Pos.*TileNum*(ipt,j), and ES.Point.RtnsCode.*TileNum*(ipt,j) are all in the same tile when they have the same value of *TileNum*. However, as used in the vertical tiling concept, these three items are separately addressable row ranges of specific columns "vertical tiles" and would require 3 separate and different DAL calls to

retrieve them (assuming no use of wild card strings to effectively turn a single DAL calling line of code into multiple DAL calls). However, when we count tiles, three point tiles with the same value of *TileNum* or j_tile are considered one tile, not three.

Governing quality metrics: Correct number must be provided

See Also: PD.TileMap.N_Rows(j_tile) ES.Align.LocVert(j_tile,j) ES.Align.Z_ECEF(j_tile, j) PD.TileMap.TileNum(j_tile)

20.4.101 PD.TileMap.PtType(j_tile) [L2,3,4,5]

Brief Definition: PD.TileMap.PtType(j_tile) indicates whether a point tile's contents are lidar points or synthetic lidar points.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* for point oriented tiles as provided by PD.TileMap.*TileNum*(j_tile) and ranges from 1 to PD.TileMap.N_Tiles, that is, over all of the tiles. However, this should only be used for values of j_tile in the point tile range.

Units/representation: Dimensionless, UInt8

Treatment/usage: Tiles of the point kind may contain only lidar points or only synthetic lidar points. If a data set has some tiles that are lidar points and some tiles that are synthetic lidar points, then GlblPtTyping =3 and it is necessary to set PD.TileMap.PtType(j_tile) to 0 for tiles containing lidar points and to set PD.TileMap.PtType(j_tile) to 1 for tiles containing synthetic lidar points.
Use PD.TileMap.PtType(j_tile) only when GlblPtTyping =3. When PD.TileMap.GlblPtTyping is either 1 or 2, it tells the story for the data set and PD.TileMap.PtType(j_tile) should not be populated.
Note that if the data set has both lidar and synthetic lidar points, they must be of a single type in a single tile. However, the dedicated tile defining metadata should not be used additionally to communicate the values of PD.TileMap.PtType(j_tile). The dedicated tile mechanism is for the result of sorted data where separation by type at the tile level is not mandated as in the case of real vs synthetic points.

For values of j_tile that lie outside of the point tile range, set PD.TileMap.PtType(j_tile) to -1.

Governing quality metrics:

See Also: PD.TileMap.GlblPtTyping PD.HiIndx.Point.Tile PD.TileMap.N_Tiles

20.4.102 PD.TileMap.PulseRange(j_tile,k) [L1,2,3,4,5]

Brief Definition: PD.TileMap.PulseRange(j_tile,k) indicates the temporal boundaries in global

	pulse numbers of the data within each tile.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* [by tile type: point, waveform, and pulse] as provided by PD.TileMap.TileNum(j_tile) and ranges from PD.LoIndx.Pulse to PD.HiIndx.Pulse, that is from the lowest value of j_tile associated with pulse data to the highest value of j_tile associated with pulse data. The index, k, =1,2
Units/representation	Dimensionless, UInt32
Treatment/usage	For pulse data tiles, PD.TileMap.PulseRange(j_tile,1) gives the pulse number for the earliest data in the j_tileth tile, and PD.TileMap.PulseRange(j_tile,2) gives the pulse number for the latest data in the j_tileth tile. No tiles containing point or waveform data will be in the range of j_tile, covered by PD.TileMap.PulseRange(j_tile,k). There are two purposes for PD.TileMap.PulseRange(j_tile,k). The first is to support finding the pulse information that goes with a point whose IPP number is given in ES.Point.PulseNum.*TileNum*(ipt,j). The second is to support finding pulse information about IPPs without regard to a point. This may be used in or prior to the creation of points-- for example, in reconstructing sensor position data from the packed integers in SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j). To support rapid extraction of the pulse information, when the IPP number is known, finding the pulse tile with that IPP is made very simple by first scanning PD.TileMap.PulseRange(j_tile,k). The k=1,2 values are searched for the case that they bracket the desired IPP value. The first tile number (value of j_tile) to meet that criterion will be the only tile number to contain data for the desired IPP and the search is ended. The global IPP values carried in PD.TileMap.PulseRange(j_tile,k) count consecutively and continuously across pulse tiles, starting with 0 for the first IPP of the collection operation. This contrasts with the IPP count resetting that is used in support of ES.Point.PulseNum.*TileNum*(ipt,j). The pulse tiles only use the IPP as an index (thus there is no data load due to carrying pulse numbers) whereas the *values* of ES.Point.PulseNum.*TileNum*(ipt,j) are pulse numbers and storage volume management may be necessary.
Governing quality metrics	
See Also	PD.TileMap.T_Range(j_tile,k) ES.Point.PulseNum.*TileNum*(ipt,j)

20.4.103 PD.TileMap.RayRange(j_tile,k) [L1,2]

Brief Definition	PD.TileMap.RayRange(j_tile,k) provides bounding values of "iray", the ray index for waveform bearing tiles.
indices	The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.*TileNum*(j_tile) and ranges from PD.LoIndx.WF to PD.HiIndx.WF, that is from the lowest value of j_tile associated with waveform data to the highest value of j_tile associated with waveform data. The index, k, =1,2
Units/representation	Dimensionless, UInt32
Treatment/usage	A ray index, "iray", is associated with a particular pairing of an array choice and a detecting element on that array. This data model supports making this mapping of element and array to ray either tile specific or global for a given data set. SC.WF.RayDesc.Arry.*TileNum*(iray), together with SC.WF.RayDesc.Elmt.*TileNum*(iray), makes an assignment of the combination of a choice of a particular array and an element on that array to a single ray index, "iray". This assignment is tile specific in the sense that a value of iray is uniquely mapped to an array and an element only within the tile identified by *TileNum*. Thus, when only a few rays have populated waveform clips, relatively low values of 'iray" may be reused across tiles, with local meaning provided via SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). In other cases, the mapping of array and element to tile is done globally for the data set via SC.WF.RayDesc.ElmtGlbl(iray) and SC.WF.RayDesc.ArryGlbl(iray). When neither SC.WF.RayDesc.ArryGlbl(iray) nor SC.WF.RayDesc.Arry.*TileNum*(iray) are populated, all rays are assumed to belong to the only array reported. Regardless of whether the mapping of array and element to specific rays is tile specific or global, a given waveform bearing tile will have some range of rays. For waveform data tiles, the ray range of a waveform oriented tile (labeled by its value of j_tile) is given by: PD.TileMap.RayRange(j_tile,1) gives the lowest iray number for the data in the j_tileth tile, and PD.TileMap.PulseRange(j_tile,2) gives the highest iray number for the data in the j_tileth tile. No tiles containing point or pulse data will be in the range of j_tile, covered by PD.TileMap.RayRange(j_tile,k). Waveform bearing tiles can be checked for ray content quickly via PD.TileMap.RayRange(j_tile,k). However, it is not usually the case that a ray appears only in one tile (especially for level-1 data, which is most naturally pulse ordered). The data would have had to have been specially

sorted for that to be the case. Thus, to find all waveforms associated with a specific ray, the tile for each row (value of j_tile) of PD.TileMap.RayRange(j_tile,k) that brackets the ray of interest should be accessed.
Furthermore, the "iray" value in a tile must be interpreted via the population of global and *TileNum* versions of SC.WF.RayDesc.Arry and SC.WF.RayDesc.Elmt. This is because the ray range indicated in PD.TileMap.RayRange(j_tile,k) could be the same for two tiles not having any rays in common.

Governing quality metrics

See Also SC.WF.RayDesc.ElmtGlbl(iray) SC.WF.RayDesc.Elmt.*TileNum*(iray) SC.WF.RayDesc.ArryGlbl(iray) SC.WF.RayDesc.Arry.*TileNum*(iray)

20.4.104 PD.TileMap.Return_Range(j_tile,k) [L2,3,4,5]

Brief Definition PD.TileMap.Return_Range(j_tile,k) provides the range of returns in the j_tileth tile.

indices The index, j_tile, corresponds to a unique evaluation of *TileNum* [within each tile category separately: points, pulses, waveforms] as provided by PD.TileMap.*TileNum*(j_tile) and ranges from 1 to PD.TileMap.N_Tiles, that is, over all of the tiles.
The index, k, =1,2

Units/representation Dimensionless, UInt8

Treatment/usage When a dedicated tile strategy is not used that makes all tiles carry a single value of the return, the range of returns in tile, j_tile, can be provided by populating PD.TileMap.Return_Range(j_tile,k).
PD.TileMap.Return_Range(j_tile,1) carries the lowest return number in the j_tileth tile and PD.TileMap.Return_Range(j_tile,2) carries the maximum return for the j_tileth tile. Note that these are the return numbers, not the returns code obtainable from ES.Point.RtnsCode.*TileNum*(ipt,j) or from ES.WFClp.RtnsCode.*TileNum*(iray, ipi).

Governing quality metrics Must have correct value.

See Also PD.TileMap.N_Tiles PD.TileMap.DedictdTile.Cnstrnt1.Val(j) ES.Point.RtnsCode.*TileNum*(ipt,j)

20.4.105 PD.TileMap.StartPulseNum.*TileNum*(j_parent, j,k) [L2,3,4,5]

Brief Definition PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) supports temporal mapping and reducing the IPP wordsize in tiles populated with IPP values

	assigned to point or waveform clip returns by tracking the global IPP count from the first IPP (indexed as 0) of the collection data set to the first IPP in each intra-tile time segment of wide band data in tile *TileNum*.
indices	Index, j_parent, runs from PD.LoIndx.Parents to PD.HiIndx.Parents, that is, over the index range of parent data sets. If PD.LoIndx.Parents is set to 1, then PD.HiIndx.Parents is the number of parent data sets. PD.LoIndx.Parents will have the same value as PD.HiIndx.Parents for enterprise product levels 1, 2 and 3. Index, j, ranges from PD.LoIndx.Tsegts.*TileNum* to PD.HiIndx.Tsegts.*TileNum*, that is, over the number of distinct internally contiguous[187] time sequences (referred to as "intra-tile time segments" in this discussion) in the lidar returns of tile *TileNum*. The segments may be disjoint to reflect "revisit" style collection modes, or may be consecutive to allow longer collection periods to be placed in a single tile without incurring a byte volume hit due to the need to double the IPP data wordsize. Index, k=1,2,3,4 *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. *TileNum* evaluates to a character string which contains purely numeric symbols for point oriented tiles. For waveform oriented tiles, there will be an alphabetic symbol, "R" embedded to separate the return from the tile sequence number. Thus, searching *TileNum* for the character, "R" is one method of determining whether a tile carries point oriented data or waveform data. Consulting PD.LoIndx.WF(j), PD.HiIndx.WF(j), and PD.TileMap.TileNum(j_tile) is a more direct method of determining the nature of the data for a *TileNum* value. Since PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) applies only to tiles associated with lidar returns, PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) will not be populated for values of *TileNum* that correspond to pulse data tiles.[188]
Units/representation	Dimensionless, UInt32
Treatment/usage	Let us define the time intervals over which the point or waveform clip returns

[187] This word is used because all the points in one of these time segments belong to a single, relatively short, time span. However, they do not have to be time ordered within the time segment. So, we avoid calling the data consecutive as that suggests a time ordering that we wish not to impose, even though it is permitted.

[188] For pulse data tiles, the pulse ranges in global IPP values will be given by PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2).

in a tile were collected to be divided into a number of intra-tile time segments. If the returns in a tile were collected in a single continuous time span, the number of intra-tile time segments could be only 1. This is typical of a high rate area mapping style mode. We use the word "could" in the previous sentence to permit the incorporation of consecutive breakups of the single time interval into intra-tile time segments, when that is desired. Reasons for that could be dealing with very high PRFs and sensor array element counts, or the desire to have video like chunking within a single tile. On the other hand, there exist collection modes where a specific geographic region is revisited to create geometric aspect diversity. Such revisits may mean that a bit of time passes between looks and counting IPPs continuously from the earliest of the tile may require a larger word size than being able to restart the local IPP count for each of several intra-tile time segments. Considering the point index, "ipt" to be a row counter in a table (refer to figure 10.2.1-4), the time segments map to consecutive spans of rows or "ipt" ranges.

For each parent and *TileNum*, the jth intra-tile time segment is described as follows:

PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,1)=

Point case: the value of the point index, "ipt" that is the lowest value of "ipt" in the intra-tile time segment.

Waveform clip case: the value of the pulse index, "ipi" that is the lowest value of "ipi" in the intra-tile time segment.

PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,2)=

Point case: the value of the point index, "ipt" that is the highest value of "ipt" in the intra-tile time segment.

Waveform clip case: the value of the pulse index, "ipi" that is the highest value of "ipi" in the intra-tile time segment.

Note that intent of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,1) and PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,2) is to bound the list of returns for a particular intra-tile time segment. All returns in the interval bounded by these items must be the totality of returns for that time segment. The returns in a block of data identified as an intra-tile time segment do not have to be time ordered, but the grouping must be temporally intact. A processor creating data to be described by PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) may need to do some arranging of the data in a tile to assure that proper time segments of data do exist as blocks of consecutively addressable data elements.

For tile, *TileNum*, the jth intra-tile time segment due to parent data set,

j_parent we provide the following[189]:
PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,3)=the global IPP value for the return within the intra-tile time segment with the earliest IPP.

For points, PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,3) will be populated based on ES.Point.PulseNum.*TileNum*(ipt, j) having the local IPP value of 0 for the earliest return in each intra-tile time segment.

PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,4)=the global IPP value of the return within the intra-tile time segment with the latest IPP.

The purpose of PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) is not only to provide a temporal tile map but to permit populated values for pulse numbers [found in ES.Point.PulseNum.*TileNum*(ipt)] within (point) tiles to count only from the start of a local sequence of IPPs, not the start of the entire data set. This "pulse offset" number would be added to the value of the pulse number on a point or waveform clip in a tile to find the actual value as if counted from the start of the collection. For this to work properly, the first IPP[190] in any point tile's intra-tile time segment would be given a local value of 0.

So if IPPs 0 to 9999 (counting from the start of collection) are in the first tile, we would set PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,3) for the first intra tile segment of the first tile to 0 and PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,3) for the first intra tile segment of the second tile to 10000. Then, the first global IPP of the second tile, for which we populate a value of ES.Point.PulseNum.*TileNum*(first value of ipt) =0, would be known to be $IPP_{(\text{tile 2, intra tile segment 1})}' = 0 +$ PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,3), which would be = 0 + 10000 =10000. The value of PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,3) for the third tile depends on the number of IPPs in the second tile. But this does not yet address any segments after the first for any tile.

Now focus on the first tile and its IPPs that number 0 to 9999. Suppose we want this to be a single intra-tile time segment. We make the range of j degenerate by setting PD.LoIndx.Tsegts.*TileNum* =

[189] This discussion proceeds with the illustration as if k=1,2,3,4. The PD.LoIndx.*TileNum*.Tsegts might actually be implemented with a first index value of 0.

[190] In a level-4 data set composed of a merger of multiple lidar data sets, we track the IPPs separately for each parent. Points having close XYZ values in a L-4 could have been collected at very different times from the start of collection, and the mapping of IPP to earthframe LOS to the target is likely to be different for each parent.

PD.HiIndx.Tsegts.*TileNum*= say 1, or 0.
Then PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,1) =1 (or 0 if the software has chosen 0-based index counting). Then PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,2) = PD.TileMap.N_Rows(j_tile) (or 1 less if the processor has chosen 0-based index counting) where j_tile is derived from the *TileNum* of ES.Point.PulseNum.*TileNum*(ipt).
PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,3)=0, and
PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,4)=9999.
Incidently, PD.TileMap.PulseRange(j_tile,1)=0 and PD.TileMap.PulseRange(j_tile,2)=9999 as well. Here j_tile is also found from the *TileNum* in ES.Point.PulseNum.*TileNum*(ipt).

Now consider a case where the first tile has 60 thousand pulses. There is a first visit to the region of the tile during pulses 0 to 29999. The second visit is during pulses 60000 to 89999. It would take a UInt32 number to represent all the pulse numbers above 65535. But we could get away with only UInt16 if we could use an offset for the second time segment. Let's say that there are 40,000,000 points in each time segment. So we are talking about 80,000,000 UInt16 numbers vs 80,000,000 UInt32 numbers, a byte volume difference of 0.16GB for a single tile. Then we would populate as follows:
The range of j would be 2.

PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,1)= 1
PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,2)= 40000000
PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,3)=0 This will have the first local IPP value of 0.
PD.TileMap.StartPulseNum.*TileNum*(j_parent,1,4)=29999
This will have a local IPP value of 29999.

PD.TileMap.StartPulseNum.*TileNum*(j_parent,2,1)= 40000001
PD.TileMap.StartPulseNum.*TileNum*(j_parent,2,2)= 80000000
PD.TileMap.StartPulseNum.*TileNum*(j_parent,2,3)= 60000 This will have the second local IPP value of 0.
PD.TileMap.StartPulseNum.*TileNum*(j_parent,2,4)=89999 This will also have a local IPP value of 29999.

By using this strategy, the IPP numbers can be kept to smaller word sizes for all point tiles. There is no need to consume the space for an integer capable of carrying the digits of the actual collection start based IPP value for each and every point or waveform. This tile referenced local pulse numbering permits 50% reduction in bits to carry the pulse numbers. Furthermore, within any single intra-tile time segment, the returns need

not be time sorted. A return with local IPP=0 can be anywhere in the intra-tile time segment.

Notice that while we can use the above mechanism to limit the number of bytes dedicated to documenting pulse numbers in tiles that have a large number of points, we have a slightly more limited practical solution for the case of waveform clips. Rather than restarting the "ip" count in each intra-tile segment in a tile, we restart the count only for the first intra-tile segment of a tile, per PD.WFClp.PulseNum.*TileNum*(ipi)[191]. That is, global IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndex.Tsegts, 3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent, PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. We have limited the number of pulses in a WFCLP tile to 65536 by making PD.WFClp.PulseNum.*TileNum*(ipi) be a UInt16. This does not seem to be a major loss of flexibility. Recall that the data for a single clip will be large compared to that for a single point, so the number of clips in a tile will be much less than the number of points if we put a limit on the byte volume for a single tile. With this strategy for global IPP recovery, the processor can still tailor the number of waveform clips in intra-tile time segments and the number of such time segments per tile to support requested exploitation objectives, collection parameters and digital data sizing/bandwidth constraints.

Governing quality metrics

See Also PD.TileMap.TileNum(j_tile) ES.Point.PulseNum.*TileNum*(ipt) PD.WFClp.PulseNum.*TileNum*(ipi) PD.HiIndx.Tsegts.*TileNum*

20.4.106 PD.TileMap.TileDelT [L1,2,3,4,5]

Brief Definition PD.TileMap.TileDelT provides the number of pulses per tile when a fixed number of pulses per tile tiling scheme is in use.

indices None

Units/representation Dimensionless, Uint16

[191] This means that PD.WFClp.PulseNum.*TileNum*(ipi) assigns sparse pulse indexing to actual pulse counting from the eariest return in the tile such that PD.TileMap.StartPulseNum.*TileNum*(j_parent,2,3), PD.TileMap.StartPulseNum.*TileNum*(j_parent,3,3) etc. will be global IPP values mapping to local IPP values corresponding to the number of pulses elapsed from the earliest return of the tile to the start of the jth intra-tile time segment block of returns. Like the point data case, we can block the waveform returns into time segments, but for waveforms we have no choice about the wordsize for the IPP number.

Treatment/usage

When PD.TileMap.TileDelT is populated, it acts as a flag to indicate that all point and pulse oriented tiles have the same number of pulses, and that the numbering of tiles is sequential in time. [That is, the sorting order of *TileNum* is ascending[192] in the value of the first IPP in each of the consecutively named tiles.] This is most appropriate in enterprise level-1 data and in higher levels when a video like 3-D image stream, likely in the sensor coordinate frame, is the main use scenario.

PD.TileMap.TileDelT is the global value of the number of IPPs of data in each tile In such a case, it would be unnecessary and ill advised to populate PD.TileMap.PulseRange(j_tile,k), since that information can be derived from the tile index value (i.e., j_tile) and PD.TileMap.TileDelT. Similarly, PD.TileMap.T_Range(j_tile,k) could be computed by converting first and last pulse times to seconds with SC.Pulse.TxCount.*Il.TileNum*(ip,j) and SC.Swath.TimeStep. When the number of IPPs per tile vary across tiles, PD.TileMap.TileDelT must not be present in the file. The tile mapping functions are then provided by PD.TileMap.PulseRange(j_tile,k) and PD.TileMap.T_Range(j_tile,k).

Governing quality metrics

See Also

PD.TileMap.T_Range(j_tile,k) PD.TileMap.PulseRange(j_tile,k)

20.4.107 PD.TileMap.TileNum(j_tile) [L1,2,3,4,5]

Brief Definition

PD.TileMap.TileNum(j_tile) provides the mapping between the ordinal index of a tile and the evaluated *TileNum* sub-strings in tags for tiled data items.

indices

The index, j_tile, ranges from 1 to PD.TileMap.N_Tiles, that is, over all of the tiles.

Units/representation

Dimensionless, BCS Character- Variable length provided by PD.Wordsize.TileNum

Treatment/usage

In support of a name-value basis for data set content, this CMMD employs the method of using evaluated sub-strings to give unique names to the multiple instantiations of a data item that might be carried in a given data set. For those instantiations which are needed to support the partitioning of the logical over-all lidar data table into "vertical tiles", the sub-string to be evaluated is *TileNum*. There are PD.TileMap.N_Tiles unique values of the character string, *TileNum*, in a lidar data set governed by this CMMD. PD.TileMap.TileNum(j_tile) establishes a unique mapping of each value of

[192] This would be a good practice whenever the constraints of the product/product usage allow this option, even if tiles do not have a fixed number of pulses.

TileNum [and, as a pair, a tile orientation from point, pulse and waveform] to an integer by being an ordered list of the separate evaluations of *TileNum*. Thus, if PD.TileMap.TileNum(j_tile) is supplied for a 10 tile data set, we might have

PD.TileMap.TileNum(1) = "00001"
PD.TileMap.TileNum(2) = "00002"
PD.TileMap.TileNum(3) = "00003"
PD.TileMap.TileNum(4) = "00004"
PD.TileMap.TileNum(5) = "00005"
PD.TileMap.TileNum(6) = "00006"
PD.TileMap.TileNum(7) = "00007"
PD.TileMap.TileNum(8) = "00008"
PD.TileMap.TileNum(9) = "1R001"
PD.TileMap.TileNum(10) = "100R2"

In the above case, PD.TileMap.TileNum establishes the integer 6 as corresponding to the TileNum string, "00006", and the integer 10 as corresponding to the TileNum string, "100R2".

An important rule must be followed to assure that the tiles can be efficiently mapped. In creating the assignment of the values of *TileNum* to an ordinal count of tiles by populating PD.TileMap.TileNum(j_tile), all point related tiles must be in a contiguous sequence of j_tile. All waveform bearing tiles must be in a separate contiguous sequence of j_tile and all pulse oriented tiles must have their own contiguous sequence of j_tile values.
As it is part of a tag for a data item, *TileNum* values must support unique instantiations of vertical tile vectors. See the level-2 specific discussion regarding *ChNum* and *TileNum* in 10.1.2.4. Also, per 10.1.2.4, the mapping of *TileNum* values to j_tile values is one to one.
When a data set has no points or no waveforms, there would be no sub-sequence of j_tile for the non-existent data bearing tiles. Substantial capability accrues to populating the pulse tiles, so those should generally be present in most lidar data data sets. So in the illustration above, values of j_tile =6,7,8 might be for pulse data bearing tiles.

Governing quality metrics: Must have correct values.

See Also: PD.Wordsize.TileNum PD.TileMap.N_Tiles

20.4.108 PD.TileMap.MGRS_Zone(j_tile) [L1, 2,3,4,5]

Brief Definition: PD.TileMap.MGRS_Zone(j_tile) provides the MGRS zone identifier associated with tile, j_tile. This is needed at enterprise L-2 and above when

	the Cartesian frame origins are chosen to match MGRS zone location by tile.
indices	The index, j_tile, corresponds to unique evaluations of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges from 1 to PD.TileMap.N_Tiles, that is, over all of the tiles.
Units/representation	Dimensionless, BCS Character, 15 characters[193]
Treatment/usage	The Military Grid Reference System (MGRS http://earth-info.nga.mil/GandG/coordsys/grids/referencesys.html) is an extension of the Universal Transverse Mercator (UTM) grid system that permits identifying zones in the UTM longitudinal bands with finer granularity than UTM and uses the Universal Polar Stereographic (UPS) system near the earth's poles. PD.TileMap.MGRS_Zone(j_tile) has been provided in order to support documenting the horizontal coverage of a tile of lidar data. Supplying the MGRS zone whose size and position most closely matches a tile's content is a convenient way of documenting tile coverage. The meaning of PD.TileMap.MGRS_Zone(j_tile) is slightly contextual, depending on the enterprise level of the product that it is populated within. • For enterprise level-1 data, which is time domain data, nominal tile ground coverage computed before the data is processed to XYZ in the earth frame is provided by PD.TileMap.MGRS_Zone(j_tile). This is discussed in 10.1.2.5. [Angle-angle-range data in the sensor frame is a possible level-2 option that would use data computed at the L-1 fidelity level to document tile coverage as well.] • For enterprise level-2 and above, points in the earth frame have been computed for waveform clips and, commonly, the points are stored in the earth frame. Here, the actual bounding points are to be selected in populating the MGRS zone coverage. It is recommended that tiled data be entirely within the stated zone and that the zone size be selected as small as possible to circumscribe the tiled LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) data. This CMMD uses only local Cartesian frames to define XYZ positions in point clouds. In this CMMD, per ES.Swath.SensorBasedCoordFlg(j), such a frame is specified either with respect to an earth frame (ECEF coordinates or geodetic coordinates referred to theWGS-84 Ellipsoid with X and Y axes aligned per ES.Align.XYZ_Flag) or a frame that co-moves with the sensor. An MGRS zone identifier can be attached to a tile of point data in either case. The meaning of ES.Align.XYZ_Flag may be linked to the value of PD.TileMap.MGRS_Zone(j_tile). The MGRS system provides a naming

[193] Accommodates the maximum length zone designator. Blank post pad as needed when larger zones are in use.

scheme for 10^5, 10^4, 10^3, 10^2, 10^1 and 10^0 –meter squares within the UTM zones. As discussed in the MD.Coverage.Bounds.NumZones paragraph, the size of an MGRS zone is determined by its name. When using MGRS to define the local XYZ frames, only a single zone is assigned to a single tile and that zone is used to determine the orientation of the axes. When ES.Origin.MGRS_Flag is set to 4, the MGRS zone may be different for each tile. In such a case, PD.TileMap.MGRS_Zone(j_tile) supplies the zone identifier for each tile.
Tiles are of variable size and so are MGRS zones. When we say there is only one MGRS zone per tile, we speak in units of the MGRS square chosen via the names in PD.TileMap.MGRS_Zone(j_tile). The fact that there may be MGRS zones defined that are smaller than what is used to populate PD.TileMap.MGRS_Zone(j_tile) should not be a source of confusion, as those zones are not part of the metadata population for a given CMMD compliant data set. The size of the MGRS square assigned to a tile should be matched to the coverage of the tile. This, in turn, depends on the point density and the volumetric drivers for selecting tile size. Appendix 10.2.1.4 gives a full discussion of the CMMD rules relating tiles, points, and MGRS zones.

Governing quality metrics

See Also: ES.Swath.SensorBasedCoordFlg(j) ES.Origin.X_MGRS(j_tile) ES.Origin.MGRS_Flag ES.Align.X_ECEF(j_tile,j) ES.Align.XYZ_Flag MD.Coverage.Bounds.NumZones

20.4.109 PD.TileMap.X_Range(j_tile,k) [L2,3,4,5]

Brief Definition: PD.TileMap.X_Range(j_Tile,k) gives the tile based Cartesian bounds in the X (easting measurement) direction when a global Cartesian frame is in use for the data set.

indices: The index, j_tile, corresponds to unique instantiations of *TileNum* as provided by PD.TileMap.*TileNum*(j_tile) and ranges from PD.LoIndx.Point.Tile to PD.HiIndx.Point.Tile, that is, over the point tiles.
Index, k, =1,2

Units/representation: Meters, float

Treatment/usage: This CMMD puts point clouds exclusively in Cartesian frames, one per tile. Multiple tiles may use the same frame, or if desired, the data set may be tiled so that there is a local geodetically aligned Cartesian frame unique to each tile. In the former cases, the Cartesian tile frames may use a common origin but different tile specific offsets in the linear expression that takes the integer X position data point and converts it via a scale factor to a distance in meters

from the origin.
PD.TileMap.X_Range(j_Tile,1) is the minimum value of X **after** any scale and offset application.
PD.TileMap.X_Range(j_Tile,2) is the maximum value of X **after** any scale and offset application.
PD.TileMap.X_Range(j_Tile,k) should only be populated when all the tile frames use a common origin and orientation, i.e. when the data set uses a global Cartesian frame. Its presence in a file becomes a flag for this strategy for the data set. Its absence prevents meaningless searches for XYZ position ranges within tiles.

Governing quality metrics

See Also: PD.TileMap.PulseRange(j_tile,k) PD.TileMap.RayRange(j_tile,k) ES.Scale.X_Fac ES.Offset.X(j_tile) LD.Point.Pos.*TileNum*(ipt,j) PD.HiIndx.Point.Tile

20.4.110 PD.TileMap.Y_Range(j_tile,k) [L2,3,4,5]

Brief Definition: PD.TileMap.Y_Range(j_Tile,k) gives the tile based Cartesian bounds in the Y (northing measurement) direction when a global Cartesian frame is in use for the data set.

indices: The index, j_tile, corresponds to unique instantiations of *TileNum* as provided by PD.TileMap.*TileNum*(j_tile) and ranges from PD.LoIndx.Point.Tile to PD.HiIndx.Point.Tile, that is, over the point tiles.
Index, k, =1,2

Units/representation: Meters, float

Treatment/usage: This CMMD puts point clouds exclusively in Cartesian frames, one per tile. Multiple tiles may use the same frame, or if desired, the data set may be tiled so that there is a local geodetically aligned Cartesian frame unique to each tile. In the former cases, the Cartesian tile frames may use a common origin but different tile specific offsets in the linear expression that takes the integer Y position data point and converts it via a scale factor to a distance in meters from the origin.
PD.TileMap.Y_Range(j_Tile,1) is the minimum value of Y **after** any scale and offset application.
PD.TileMap.Y_Range(j_Tile,2) is the maximum value of Y **after** any scale and offset application.
PD.TileMap.Y_Range(j_Tile,k) should only be populated when all the tile frames use a common origin and orientation, i.e. when the data set uses a global Cartesian frame. Its presence in a file becomes a flag for this strategy for the data set. Its absence prevents meaningless searches for XYZ position

ranges within tiles.

Governing quality metrics

See Also PD.TileMap.PulseRange(j_tile,k) PD.TileMap.RayRange(j_tile,k) ES.Scale.Y_Fac ES.Offset.Y(j_tile) LD.Point.Pos.*TileNum*(ipt,j) PD.HiIndx.Point.Tile

20.4.111 PD.TileMap.Z_Range(j_tile,k) [L2,3,4,5]

Brief Definition PD.TileMap.Z_Range(j_Tile,k) gives the tile based Cartesian bounds in the Z (vertical measurement) direction when a global Cartesian frame is in use for the data set.

indices The index, j_tile, corresponds to unique instantiations of *TileNum* as provided by PD.TileMap.*TileNum*(j_tile) and ranges from PD.LoIndx.Point.Tile to PD.HiIndx.Point.Tile, that is, over the point tiles.
Index, k, =1,2

Units/representation Meters, float

Treatment/usage This CMMD puts point clouds exclusively in Cartesian frames, one per tile. Multiple tiles may use the same frame, or if desired, the data set may be tiled so that there is a local geodetically aligned Cartesian frame unique to each tile. In the former cases, the Cartesian tile frames may use a common origin but different tile specific offsets in the linear expression that takes the integer Z position data point and converts it via a scale factor to a distance in meters from the origin.
PD.TileMap.Z_Range(j_Tile,1) is the minimum value of Z **after** any scale and offset application.
PD.TileMap.Z_Range(j_Tile,2) is the maximum value of Z **after** any scale and offset application.
PD.TileMap.Z_Range(j_Tile,k) should only be populated when all the tile frames use a common origin and orientation, i.e. when the data set uses a global Cartesian frame. Its presence in a file becomes a flag for this strategy for the data set. Its absence prevents meaningless searches for XYZ position ranges within tiles.

Governing quality metrics

See Also PD.TileMap.PulseRange(j_tile,k) PD.TileMap.RayRange(j_tile,k) ES.Scale.Z_Fac ES.Offset.Z(j_tile) LD.Point.Pos.*TileNum*(ipt,j) PD.HiIndx.Point.Tile

20.4.112 PD.TileMap.T_Range(j_tile,k) [L1,2,3,4,5]

Brief Definition PD.TileMap.T_Range(j_tile,k) gives the temporal range of each tile of data in

	the data set.
indices	The index, j_tile, corresponds to a unique instantiation of *TileNum* as provided by PD.TileMap.*TileNum*(j_tile) and ranges from 1 to PD.TileMap.N_Tiles, that is, over all of the tiles. Index, k, =1,2
Units/representation	Seconds, double precision float
Treatment/usage	Lidar data is collected over an operation that has a duration in time, often lasting many minutes. There is a natural pulse ordering when data is processed and not subsequently sorted. Also, if ground positions are sampled sequentially, there tends to be a high degree of time ordering to data that is sorted by horizontal position. When points or waveforms are tiled, there will be a time interval associated with each tile. Furthermore, this is trivially true for pulse data tiles. PD.TileMap.T_Range(j_tile,1) is the time in seconds from the start of the collection associated with j_tile for the earliest data in the j_tileth tile. PD.TileMap.T_Range(j_tile,2) is the time in seconds from the start of the collection associated with j_tile for the latest data in the j_tileth tile. Note that if tiles are dedicated by parent in a level-4 product, a base time can be associated with each tile through the PD.TileMap.Dedictd.* data. which maps to the parent data set.
Governing quality metrics	
See Also	PD.TileMap.DedictdTile.Cnstrnt2.Val(j) PD.TileMap.DedictdTile.TileNumValLow(j) PD.TileMap.DedictdTile.TileNumValHi(j) PD.TileMap.PulseRange(j_tile,k) PD.TileMap.Y_Range(j_Tile,k)

20.4.113 PD.TileMap.WF_Returns(ich,j) [L2]

Brief Definition	PD.TileMap.WF_Returns(ich,j) gives the number of tiles at each waveform return level for each channel.
indices	Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m), that is, over the channels of the current data set. Index, j, ranges from PD.LoIndx.WF(2) to PD.HiIndx.WF(2) that is nominally from 1 to the highest number of wave clip returns on any pulse ray for a given channel.
Units/representation	Dimensionless, UInt16
Treatment/usage	Note that the per the discussion of 10.2.1.2, the return number for a waveform

	clip is encoded into the value of *TileNum* for tiles bearing waveform clips when the data set also has point bearing tiles. To understand a data set without having to parse all of the *TileNum* strings, PD.TileMap.WF_Returns(ich,j) is defined to carry the number of tiles of data from channel 'ich' with the number of waveform returns =value of j index, which ranges from 1 to the max number of waveform clips for any pulse-ray present for the channel ich. So if WF_Return(2,3)=25, that means that there are 25 tiles with 3rd returns for channel #2.
Governing quality metrics	Must be the correct value
See Also	LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) PD.HiIndx.WF(j)

PD.IntTileMap.*

PD.IntTileMap.* flags support the reduction of repeated data items in tiles via an encoding.

20.4.114 PD.IntTileMap.RtnsCode.*TileNum*.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.RtnsCode.*TileNum*.Flg indicates the actual number of rows in the tiled data array [either ES.Point.RtnsCode.*TileNum*(ipt,j) or ES.WFClp.RtnsCode.*TileNum*(ip,iray,j)] when repetition factors are used for returns code data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. This flag determines whether repetition factors are in use for ES.Point.RtnsCode.*TileNum*(ipt,j) and ES.WFClp.RtnsCode.*TileNum*(ip,iray,j), by separate value of *TileNum*. The same value of *TileNum* will never appear for both the point and WFClp returns code data due to the naming convention in 10.2.1.2.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	ES.Point.RtnsCode.*TileNum*(ipt,j) ES.WFClp.RtnsCode.*TileNum*(ip,iray,j)

20.4.115 PD.IntTileMap.PulseNum.*TileNum*.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.PulseNum.*TileNum*.Flg indicates the actual number of rows in the tiled data array, ES.Point.PulseNum.*TileNum*(ipt,j), when repetition factors are used for pulse number data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. This flag determines whether repetition factors are in use for ES.Point.PulseNum.*TileNum*(ipt,j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	ES.Point.PulseNum.*TileNum*(ipt, j)

20.4.116 PD.IntTileMap.ArrayNum.*TileNum*.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.ArrayNum.*TileNum*.Flg indicates the actual number of rows in the tiled data array, PF.Point.IrayNum.*TileNum*(ipt,j), when repetition factors are used for array number data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index

	has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. This flag determines whether repetition factors are in use for PF.Point.IrayNum.*TileNum*(ipt,j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	PF.Point.IrayNum.*TileNum*(ipt,j)

20.4.117 PD.IntTileMap.ChNum.*TileNum*.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.ChNum.*TileNum*.Flg indicates the actual number of rows in the tiled data array, ES.Point.ChNum.*TileNum*(ipt,j), when repetition factors are used for channel number data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. This flag[194] determines whether repetition factors are in use for ES.Point.ChNum.*TileNum*(ipt,j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	ES.Point.ChNum.*TileNum*(ipt,j)

20.4.118 PD.IntTileMap.DirCase.*TileNum*.Flg [L1,2]

Brief Definition	PD.IntTileMap.DirCase.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Array.DirCaseNum.*TileNum*(ip,j), when repetition factors are used for scan phase data in the tile indicated by the value of *TileNum*.
indices	None
Units/representation	Dimensionless, UInt16

[194] Not likely to be used since the more likely channelization treatment will be separate tiles for different channels.

Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.DirCase.*TileNum*.Flg determines whether repetition factors are in use for SC.Array.DirCaseNum.*TileNum*(ip,j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	SC.Array.DirCaseNum.*TileNum*(ip,j)

20.4.119 PD.IntTileMap.PolCode.*TileNum*.Flg [L1,2,3,4,5]

Brief Definition	PD.IntTileMap.PolCode.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.PolCode.*TileNum*(ip,j), when repetition factors are used for polarization code data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.PolCode.TileNum.Flg determines whether repetition factors are in use for SC.Pulse.PolCode.*TileNum*(ip,j)[195] by separate value of *TileNum*.

[195] This would seem to require sorting data by polarization, then by pulse. This is a filter wheel scenario.

Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	SC.Pulse.PolCode.*TileNum*(ip,j)

20.4.120 PD.IntTileMap.FilterCode.*TileNum*.Flg [L1,2,3,4,5]

Brief Definition	PD.IntTileMap.FilterCode.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.FilterCode.*TileNum*(ip,j), when repetition factors are used for filter code data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.FilterCode.*TileNum*.Flg determines whether repetition factors are in use for SC.Pulse.FilterCode.*TileNum*(ip,j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	SC.Pulse.FilterCode.*TileNum*(ip,j)

20.4.121 PD.IntTileMap.RecordStrt.*TileNum*.Flg [L1,2]

Brief Definition	PD.IntTileMap.RecordStrt.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.RecordStrt.*TileNum*(ip,j), when repetition factors are used for recording start data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a

PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.RecordStrt.*TileNum*.Flg determines whether repetition factors are in use for SC.Pulse.RecordStrt.*TileNum*(ip,j) by separate value of *TileNum*.

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: SC.Pulse.RecordStrt.*TileNum*(ip,j)

20.4.122 PD.IntTileMap.Scan.Num.*TileNum*.Flg [L1,2]

Brief Definition: PD.IntTileMap.Scan.Num.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.Scan.Num.*TileNum*(ip,j), when repetition factors are used for scan number data in the tile indicated by the value of *TileNum*.

indices: None

Units/representation: Dimensionless, UInt16

Treatment/usage: Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.Scan.Num.*TileNum*.Flg determines whether repetition factors are in use for SC.Pulse.Scan.Num.*TileNum*(ip,j) by separate value of *TileNum*.

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: SC.Pulse.Scan.Num.*TileNum*(ip,j)

20.4.123 PD.IntTileMap.Scan.Phase.*TileNum*.Flg [L1,2]

Brief Definition: PD.IntTileMap.Scan.Phase.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.Scan.Phase.*TileNum*(ip,j), when repetition factors are used for scan phase data in the tile indicated by the value of

	TileNum.
indices	None
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.Scan.Phase.*TileNum*.Flg determines whether repetition factors are in use for SC.Pulse.Scan.Phase.*TileNum*(ip,j)[196] by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	SC.Pulse.Scan.Phase.*TileNum*(ip,j)

20.4.124 PD.IntTileMap.NadirClass.*ChNum*.Flg [L1,2,3,4,5]

Brief Definition	PD.IntTileMap.NadirClass.*ChNum*.Flg indicates the actual number of rows in the array, SC.Scan.NadirClass.*ChNum*(is), when repeat factors are used.
indices	None
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding data item is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that SC.Scan.NadirClass.*ChNum*(is) is viewed, in this discussion, as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.NadirClass.*ChNum*.Flg = 0, the value of the "is" index (which we call "j" when using repeats) ranges over the actual number of scans in the data set. When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding data array has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in

[196] This is a very unlikely case. Scenario is most likely a system diagnostic file that is sorted by scan phase.

the populated data vector is reduced to PD.IntTileMap.NadirClass.*ChNum*.Flg. The logical row for SC.Scan.NadirClass.*ChNum*(is) determines the actual scan being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.NadirClass.*ChNum*.Flg is greater than zero, the repetition factors for each value in the jth row of PD.IntTileMap.NadirClass.*ChNum*.Flg is provided in PD.IntTileMap.NadirClass.*ChNum*.Reps(j).

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: SC.Scan.NadirClass.*ChNum*(is) PD.IntTileMap.NadirClass.*ChNum*.Reps(j).

20.4.125 PD.IntTileMap.NadirClass.*ChNum*.Reps(j) [L1,2,3,4,5]

Brief Definition: PD.IntTileMap.NadirClass.*ChNum*.Reps(j) indicates the number of times the jth unique scan nadir class value in SC.Scan.NadirClass.*ChNum*(is) is repeated in consecutive scans.

indices: Index, j, ranges from 1 to PD.IntTileMap.NadirClass.*ChNum*.Flg, that is over the actual number of rows in the array, SC.Scan.NadirClass.*ChNum*(is), when repeat factors are used.

Units/representation: Dimensionless, UInt32

Treatment/usage: PD.IntTileMap.NadirClass.*ChNum*.Reps(j) should only be populated when PD.IntTileMap.NadirClass.*ChNum*.Flg >0.

When populated, PD.IntTileMap.NadirClass.*ChNum*.Reps(j) indicates the number of times the jth sequentially unique[197] value in SC.Scan.NadirClass.*ChNum*(is) is repeated. When repeat factors are in use, the index range of "j" in SC.Scan.NadirClass.*ChNum*(j) is also from 1 to PD.IntTileMap.NadirClass.*ChNum*.Flg. That is, PD.IntTileMap.NadirClass.*ChNum*.Reps(j), is the key information needed to determine which element of SC.Scan.NadirClass.*ChNum*(j) is the scan nadir class value for a specific scan. When the repeats are cumulatively counted, the value of the scan nadir class for an actual scan number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.NadirClass.*ChNum*.Flg.

Governing quality metrics: At least one value of PD.IntTileMap.NadirClass.*ChNum*.Reps(j) should have value >>1 when the repetition factor strategy is in use for scan nadir classes.

[197] If a value appears more than once, but with a diffent value (or values) intervening, it is treated as unique each time it reappears in the sequence and has it own repeat factor for each sequence of logical rows that it appears in.

See Also	SC.Scan.NadirClass.*ChNum*(is) PD.IntTileMap.NadirClass.*ChNum*.Flg

20.4.126 PD.IntTileMap.Origin.E_X.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.Origin.E_X.Flg indicates the actual number of rows in the array, ES.Origin.X_ECEF(j_tile) when repeat factors are used.
indices	None
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that ES.Origin.X_ECEF(j_tile) is viewed, in this discussion, as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.Origin.E_X.Flg = 0, the value of the j_tile index ranges over actual number of tiles in the data set. When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in the populated data vector is reduced to PD.IntTileMap.Origin.E_X.Flg. The logical row for ES.Origin.X_ECEF(j_tile) determines the actual tile being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.Origin.E_X.Flg is greater than zero, the repetition factor for each value in the jth row of ES.Origin.X_ECEF(j,tile) is provided in PD.IntTileMap.Origin.E_X.Reps(j).
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	ES.Origin.X_ECEF(j_tile) PD.IntTileMap.Origin.E_X.Reps(j)

20.4.127 PD.IntTileMap.Origin.E_Y.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.Origin.E_Y.Flg indicates the actual number of rows in the array, ES.Origin.Y_ECEF(j_tile) when repeat factors are used.
indices	None
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that ES.Origin.Y_ECEF(j_tile) is viewed, in this discussion, as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.Origin.E_Y.Flg = 0, the value of the j_tile index ranges over actual number of tiles in the data set.

When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in the populated data vector is reduced to PD.IntTileMap.Origin.E_Y.Flg. The logical row for ES.Origin.Y_ECEF(j_tile) determines the actual tile being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.Origin.E_Y.Flg is greater than zero, the repetition factor for each value in the jth row of ES.Origin.Y_ECEF(j_tile) is provided in PD.IntTileMap.Origin.E_Y.Reps(j).

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: ES.Origin.Y_ECEF(j_tile) PD.IntTileMap.Origin.E_Y.Reps(j)

20.4.128 PD.IntTileMap.Origin.E_Z.Flg [L2,3,4,5]

Brief Definition: PD.IntTileMap.Origin.E_Z.Flg indicates the actual number of rows in the array, ES.Origin.Z_ECEF(j_tile) when repeat factors are used.

indices: None

Units/representation: Dimensionless, UInt16

Treatment/usage: Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that ES.Origin.Z_ECEF(j_tile) is viewed, in this discussion, as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.Origin.E_Z.Flg = 0, the value of the j_tile index ranges over actual number of tiles in the data set.

When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in the populated data vector is reduced to PD.IntTileMap.Origin.E_Z.Flg. The logical row for ES.Origin.Z_ECEF(j_tile) determines the actual tile being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.Origin.E_Z.Flg is greater than zero, the repetition factor for each value in the jth row of ES.Origin.Z_ECEF(j_tile) is provided in PD.IntTileMap.Origin.E_Z.Reps(j).

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: ES.Origin.Z_ECEF(j_tile) PD.IntTileMap.Origin.E_Z.Reps(j)

20.4.129 PD.IntTileMap.Origin.E_X.Reps(j) [L2,3,4,5]

Brief Definition: PD.IntTileMap.Origin.E_X.Reps(j) indicates the number of times the jth

	unique ECEF referenced origin X coordinate value is repeated in consecutive tiles.
indices	Index, j, ranges from 1 to PD.IntTileMap.Origin.E_X.Flg, that is over the actual number of rows in the array, ES.Origin.X_ECEF(j_tile) when repeat factors are used.
Units/representation	Dimensionless, UInt32
Treatment/usage	PD.IntTileMap.Origin.E_X.Reps(j) should only be populated when PD.IntTileMap.Origin.E_X.Flg >0. When populated, PD.IntTileMap.Origin.E_X.Reps(j) indicates the number of times the jth sequentially unique[197] value in ES.Origin.X_ECEF(j_tile) is repeated. When repeat factors are in use, the index range of "j_tile" in ES.Origin.X_ECEF(j_tile) is also from 1 to PD.IntTileMap.Origin.E_X.Flg. That is, PD.IntTileMap.Origin.E_X.Reps(j), is the key information needed to determine which element of ES.Origin.X_ECEF(j_tile) is the origin X coordinate value for a specific tile. When the repeats are cumulatively counted, the value of the X component of the tile origin in ECEF for an actual tile number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.Origin.E_X.Flg.
Governing quality metrics	At least one value of PD.IntTileMap.Origin.E_X.Reps(j) should have value >>1 when the repetition factor strategy is in use for tile origin coordinates.
See Also	ES.Origin.X_ECEF(j_tile) PD.IntTileMap.Origin.E_X.Flg

20.4.130 PD.IntTileMap.Origin.E_Y.Reps(j) [L2,3,4,5]

Brief Definition	PD.IntTileMap.Origin.E_Y.Reps(j) indicates the number of times the jth unique ECEF referenced origin Y coordinate value is repeated in consecutive tiles.
indices	Index, j, ranges from 1 to PD.IntTileMap.Origin.E_Y.Flg, that is over the actual number of rows in the array, ES.Origin.Y_ECEF(j_tile) when repeat factors are used.
Units/representation	Dimensionless, UInt32
Treatment/usage	PD.IntTileMap.Origin.E_Y.Reps(j) should only be populated when PD.IntTileMap.Origin.E_Y.Flg >0. When populated, PD.IntTileMap.Origin.E_Y.Reps(j) indicates the number of times the jth sequentially unique value in ES.Origin.Y_ECEF(j_tile) is repeated. When repeat factors are in use, the index range of "j_tile" in ES.Origin.Y_ECEF(j_tile) is also from 1 to PD.IntTileMap.Origin.E_Y.Flg. That is, PD.IntTileMap.Origin.E_Y.Reps(j), is the key information needed to determine which element of ES.Origin.Y_ECEF(j_tile) is the origin Y coordinate value for a specific tile. When the repeats are cumulatively

counted, the value of the Y component of the tile origin in ECEF for an actual tile number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.Origin.E_Y.Flg.

Governing quality metrics: At least one value of PD.IntTileMap.Origin.E_Y.Reps(j) should have value >>1 when the repetition factor strategy is in use for tile origin coordinates.

See Also: ES.Origin.Y_ECEF(j_tile) PD.IntTileMap.Origin.E_Y.Flg

20.4.131 PD.IntTileMap.Origin.E_Z.Reps(j) [L2,3,4,5]

Brief Definition: PD.IntTileMap.Origin.E_Z.Reps(j) indicates the number of times the jth unique ECEF referenced origin Z coordinate value is repeated in consecutive tiles.

indices: Index, j, ranges from 1 to PD.IntTileMap.Origin.E_Z.Flg, that is over the actual number of rows in the array, ES.Origin.Z_ECEF(j_tile) when repeat factors are used.

Units/representation: Dimensionless, UInt32

Treatment/usage: PD.IntTileMap.Origin.E_Z.Reps(j) should only be populated when PD.IntTileMap.Origin.E_Z.Flg >0.

When populated, PD.IntTileMap.Origin.E_Z.Reps(j) indicates the number of times the jth sequentially unique value in ES.Origin.Z_ECEF(j_tile) is repeated. When repeat factors are in use, the index range of "j_tile" in ES.Origin.Z_ECEF(j_tile) is also from 1 to PD.IntTileMap.Origin.E_Z.Flg. That is, PD.IntTileMap.Origin.E_Z.Reps(j), is the key information needed to determine which element of ES.Origin.Z_ECEF(j_tile) is the origin Z coordinate value for a specific tile. When the repeats are cumulatively counted, the value of the Z component of the tile origin in ECEF for an actual tile number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.Origin.E_Z.Flg.

Governing quality metrics: At least one value of PD.IntTileMap.Origin.E_Z.Reps(j) should have value >>1 when the repetition factor strategy is in use for tile origin coordinates.

See Also: ES.Origin.Z_ECEF(j_tile) PD.IntTileMap.Origin.E_Z.Flg

20.4.132 PD.IntTileMap.Origin.Z_X.Flg [L2,3,4,5]

Brief Definition: PD.IntTileMap.Origin.Z_X.Flg indicates the actual number of rows in the array, ES.Origin.X_MGRS(j_tile) when repeat factors are used.

indices: None

Units/representation: Dimensionless, UInt16

Treatment/usage: Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that ES.Origin.X_MGRS(j_tile) is viewed, in this discussion,

as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.Origin.Z_X.Flg = 0, the value of the j_tile index ranges over actual number of tiles in the data set.

When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in the populated data vector is reduced to PD.IntTileMap.Origin.Z_X.Flg. The logical row for ES.Origin.X_MGRS(j_tile) determines the actual tile being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.Origin.Z_X.Flg is greater than zero, the repetition factors for each value in the jth row of ES.Origin.X_MGRS(j_tile) are provided in PD.IntTileMap.Origin.Z_X.Reps(j).

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: ES.Origin.X_MGRS(j_tile) PD.IntTileMap.Origin.Z_X.Reps(j)

20.4.133 PD.IntTileMap.Origin.Z_Y.Flg [L2,3,4,5]

Brief Definition: PD.IntTileMap.Origin.Z_Y.Flg indicates the actual number of rows in the array, ES.Origin.Y_MGRS(j_tile) when repeat factors are used.

indices: None

Units/representation: Dimensionless, UInt16

Treatment/usage: Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that ES.Origin.Y_MGRS(j_tile) is viewed, in this discussion, as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.Origin.Z_Y.Flg = 0, the value of the j_tile index ranges over actual number of tiles in the data set.

When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in the populated data vector is reduced to PD.IntTileMap.Origin.Z_Y.Flg. The logical row for ES.Origin.Y_MGRS(j_tile) determines the actual tile being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.Origin.Z_Y.Flg is greater than zero, the repetition factors for each value in the jth row of ES.Origin.Y_MGRS(j_tile) are provided in PD.IntTileMap.Origin.Z_Y.Reps(j).

Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	ES.Origin.Y_MGRS(j_tile) PD.IntTileMap.Origin.Z_Y.Reps(j)

20.4.134 PD.IntTileMap.Origin.Z_Z.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.Origin.Z_Z.Flg indicates the actual number of rows in the array, ES.Origin.Z_MGRS(j_tile) when repeat factors are used.
indices	None
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that ES.Origin.Z_MGRS(j_tile) is viewed, in this discussion, as a column data vector, with each element having its own row. In this case, when PD.IntTileMap.Origin.Z_Z.Flg = 0, the value of the j_tile index ranges over actual number of tiles in the data set. When a PD.IntTileMap.xxx.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each sequentially unique value of the quantity carried in the tile. Thus, the actual number of rows in the populated data vector is reduced to PD.IntTileMap.Origin.Z_Z.Flg. The logical row for ES.Origin.Z_MGRS(j_tile) determines the actual tile being referenced and must be computed by accumulating the values of the repeat factors as described in 10.2.1.5. When PD.IntTileMap.Origin.Z_Z.Flg is greater than zero, the repetition factors for each value in the jth row of ES.Origin.Z_MGRS(j_tile) are provided in PD.IntTileMap.Origin.Z_Z.Reps(j).
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	ES.Origin.Z_MGRS(j_tile) PD.IntTileMap.Origin.Z_Z.Reps(j)

20.4.135 PD.IntTileMap.Origin.Z_X.Reps(j) [L2,3,4,5]

Brief Definition	PD.IntTileMap.Origin.Z_X.Reps(j) indicates the number of times the jth unique zone referenced origin X coordinate value is repeated in consecutive tiles.
indices	Index, j, ranges from 1 to PD.IntTileMap.Origin.Z_X.Flg, that is over the actual number of rows in the array, ES.Origin.X_MGRS(j_tile) when repeat factors are used.
Units/representation	Dimensionless, UInt32
Treatment/usage	PD.IntTileMap.Origin.Z_X.Reps(j) should only be populated when PD.IntTileMap.Origin.Z_X.Flg >0.

When populated, PD.IntTileMap.Origin.Z_X.Reps(j) indicates the number of times the jth sequentially unique value in ES.Origin.X_MGRS(j_tile) is repeated. When repeat factors are in use, the index range of "j_tile" in ES.Origin.X_MGRS(j_tile) is also from 1 to PD.IntTileMap.Origin.Z_X.Flg. That is, PD.IntTileMap.Origin.Z_X.Reps(j), is the key information needed to determine which element of ES.Origin.X_MGRS(j_tile) is the origin X coordinate value for a specific tile. When the repeats are cumulatively counted, the value of the X component of the tile origin in the tile's zone reference for an actual tile number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.Origin.Z_X.Flg.

Governing quality metrics: At least one value of PD.IntTileMap.Origin.Z_X.Reps(j) should have value >>1 when the repetition factor strategy is in use for tile origin coordinates.

See Also: ES.Origin.X_MGRS(j_tile) PD.IntTileMap.Origin.Z_X.Flg

20.4.136 PD.IntTileMap.Origin.Z_Y.Reps(j) [L2,3,4,5]

Brief Definition: PD.IntTileMap.Origin.Z_Y.Reps(j) indicates the number of times the jth unique zone referenced origin Y coordinate value is repeated in consecutive tiles.

indices: Index, j, ranges from 1 to PD.IntTileMap.Origin.Z_Y.Flg, that is over the actual number of rows in the array, ES.Origin.Y_MGRS(j_tile) when repeat factors are used.

Units/representation: Dimensionless, UInt32

Treatment/usage: PD.IntTileMap.Origin.Z_Y.Reps(j) should only be populated when PD.IntTileMap.Origin.Z_Y.Flg >0.

When populated, PD.IntTileMap.Origin.Z_Y.Reps(j) indicates the number of times the jth sequentially unique[197] value in ES.Origin.Y_MGRS(j_tile) is repeated. When repeat factors are in use, the index range of "j_tile" in ES.Origin.Y_MGRS(j_tile) is also from 1 to PD.IntTileMap.Origin.Z_Y.Flg. That is, PD.IntTileMap.Origin.Z_Y.Reps(j), is the key information needed to determine which element of ES.Origin.Y_MGRS(j_tile) is the origin Y coordinate value for a specific tile. When the repeats are cumulatively counted, the value of the Y component of the tile origin in the tile's zone reference for an actual tile number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.Origin.Z_Y.Flg.

Governing quality metrics: At least one value of PD.IntTileMap.Origin.Z_Y.Reps(j) should have value >>1 when the repetition factor strategy is in use for tile origin coordinates.

See Also: ES.Origin.Y_MGRS(j_tile) PD.IntTileMap.Origin.Z_Y.Flg

20.4.137 PD.IntTileMap.Origin.Z_Z.Reps(j) [L2,3,4,5]

Brief Definition	PD.IntTileMap.Origin.Z_Z.Reps(j) indicates the number of times the jth unique zone referenced origin Z coordinate value is repeated in consecutive tiles.
indices	Index, j, ranges from 1 to PD.IntTileMap.Origin.Z_Z.Flg, that is over the actual number of rows in the array, ES.Origin.Z_MGRS(j_tile) when repeat factors are used.
Units/representation	Dimensionless, UInt32
Treatment/usage	PD.IntTileMap.Origin.Z_Z.Reps(j) should only be populated when PD.IntTileMap.Origin.Z_Z.Flg >0. When populated, PD.IntTileMap.Origin.Z_Z.Reps(j) indicates the number of times the jth sequentially unique[197] value in ES.Origin.Z_MGRS(j_tile) is repeated. When repeat factors are in use, the index range of "j_tile" in ES.Origin.Z_MGRS(j_tile) is also from 1 to PD.IntTileMap.Origin.Z_Z.Flg. That is, PD.IntTileMap.Origin.Z_Z.Reps(j), is the key information needed to determine which element of ES.Origin.Z_MGRS(j_tile) is the origin Z coordinate value for a specific tile. When the repeats are cumulatively counted, the value of the Z component of the tile origin in the tile's zone reference for an actual tile number can be determined per the detailed discussion of 10.2.1.5. See also the discussion in PD.IntTileMap.Origin.Z_Z.Flg.
Governing quality metrics	At least one value of PD.IntTileMap.Origin.Z_Z.Reps(j) should have value >>1 when the repetition factor strategy is in use for tile origin coordinates.
See Also	ES.Origin.Z_MGRS(j_tile) PD.IntTileMap.Origin.Z_Z.Flg

20.4.138 PD.IntTileMap.Pos.*TileNum*.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.Pos.*TileNum*.Flg indicates the actual number of rows in the tiled data array, LD.Point.Pos.*ChNum*.*TileNum*(ipt,j), when repetition factors are used for position data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the

	repetition factor, and the second is the repeated value. PD.IntTileMap.Pos.*TileNum*.Flg determines whether repetition factors are in use for LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	LD.Point.Pos.*ChNum*.*TileNum*(ipt,j)

20.4.139 PD.IntTileMap.Intensity.*TileNum*.Flg [L2,3,4,5]

Brief Definition	PD.IntTileMap.Intensity.*TileNum*.Flg indicates the actual number of rows in the tiled data array, LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j), when repetition factors are used for intensity data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.Intensity.*TileNum*.Flg determines whether repetition factors are in use for LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) by separate value of *TileNum*.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	LD.Point.Intensity.*ChNum*.*TileNum*(ipt,j)

20.4.140 PD.IntTileMap.Nsamps.*TileNum*.Flg [L1,2]

Brief Definition	PD.IntTileMap.Nsamps.*TileNum*.Flg indicates the actual number of rows in the tiled data array, PD.WFClp.Nsamps.*TileNum*(iray/irays,ipi/j), when repetition factors are used for waveform data vector length data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set

or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value.
PD.IntTileMap.Nsamps.*TileNum*.Flg determines whether repetition factors are in use for PD.WFClp.Nsamps.*TileNum*(iray/irays,ipi/j) by separate value of *TileNum*.

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: PD.WFClp.Nsamps.*TileNum*(iray/irays, ipi/j) SC.WF.Rx.Nsamps(ich)

20.4.141 PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg [L1,2]

Brief Definition: PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k), when repetition factors are used for detector array element sensitivity data.

indices: None. However:
The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the receive intensity correction factor data array in the data tag.
TileNum, though part of a tag name, also acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Dimensionless, UInt32

Treatment/usage: Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is not populated or is set to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing indices (rightmost, due to CMMD row major convention) can be considered to carry a sub-array, each instantiation of which is a "row" (more literally a "layer" or "blade") in the full 3-D array. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated sub-array value of the quantity carried in the tile.
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) permits documenting the sensitivity of detecting elements of a channel as function of time through the

ip (pulse) index. When the array element sensitivities are unchanged for a number of pulses, the information would be repeated for each value of "ip". The internal tile mapping procedure indicated to be in place by the non-zero value of PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg provides only the number of actual distinct values of the full element sensitivity matrix for a *TileNum* and channel. PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg is that number.

Governing quality metrics

See Also

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)
PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum* (j)

20.4.142 PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(j) [L1,2]

Brief Definition

PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(j) provides the number of repeats for the jth sub-array sequence of unique values of the element-sensitivity item data matrix, SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k), when internal tile mapping is used to reduce redundancy due to values not changing over extended pulse sequences.

indices

The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the receive intensity correction factor data array in the data tag.

TileNum, though part of a tag name, also acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. *TileNum* acts in pulse oriented tiles such as SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) to enable blocking of data into pulse ranges.

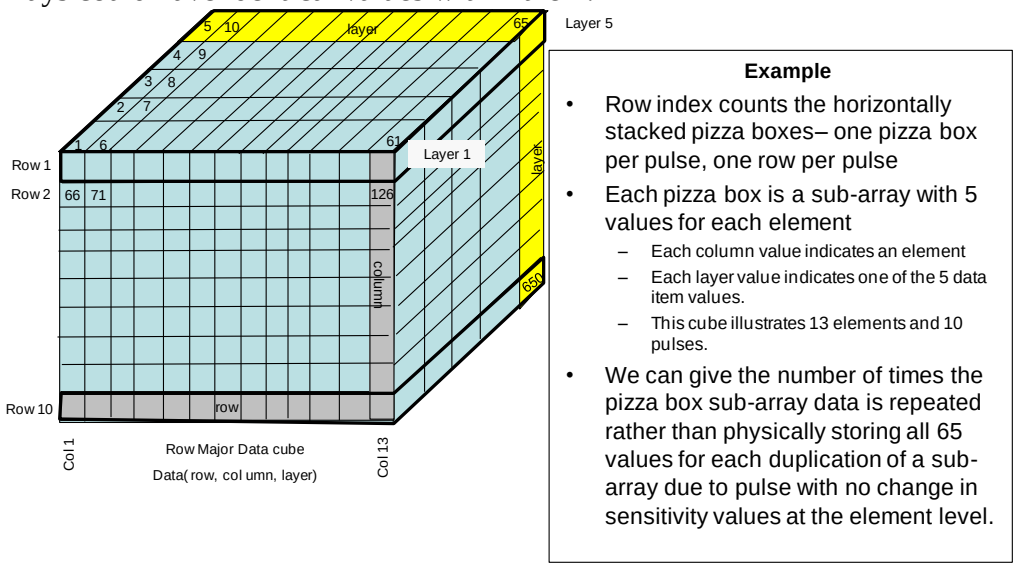
Index, j, nominally ranges from 1 to PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg, that is, over the individual unique values of the repeated sub-arrays[198]. It is permitted that software code

[198] An instantaneous element sensitivity matrix is a 2-D sub array of SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k). This 3-D matrix, providable for each tile and channel, provides a 2-D matrix for each "row" (first index, "ip", value) . The rows correspond to pulse numbers. Each "row" carries a copy of the matrix. This is a horizontal "pizza box" of consecutively stored data in the diagram, below. We expect that many consecutive rows will share a single value of the sensitivity matrix. So we wish to note the number of times such a pizza box is repeated in each consecutive run of row values.
Note that the 2 dimensions of the sensititvity matrix do NOT correspond to the rows and columns of the focal plane array. The focal plane element indices have been linearized to a single index in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k). In the illustration there are 13 elements in the FP array. One

implementations could have j vary from 0 to (D.IntTileMap.Snstvty.*ChNum.TileNum*.Flg-1) or similar alternative at the convenience of the software developers.

Units/representation Dimensionless, UInt16

Treatment/usage Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing indices (rightmost, due to CMMD row major convention) can cover sub-arrays whose values are repeated in the logical data representation. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated block of values of the quantity carried in the tile. PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(j) provides the repetition factors for the jth consecutive unique value of the data sub-array of the (number of elements) x 5 (the number of values of k, which counts data items for each element). The word "consecutive" is used to indicate that if a value of the sub-array is repeated after an intervening different value, we must restate the value to be repeated. Thus, all values of the sub-array that differ by j value of only 1 are distinct, but if the difference in j value is more than 1, the sub-arrays could have identical values within them.



If we use the internal tile mapping feature for SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k), then the number of actual

---

dimension of a 2-D sub-array tracks the element. The second dimension of the sub-array carries a set of calibration values for each element.

rows in the populated data item will be the number of unique consecutive values of the pizza box sub-arrays, as in the diagram above. This number is provided in PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg. The number of consecutive repeats for the jth pizza box sub-array is given by PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(j). If the logical cube covers 1000 pulses, but there is only one sub-array of element values repeated 1000 times, we can build a real populated data array that is only the single unique element sub-array and set PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg=1. Then PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(j) would have j ranging from 1 to 1 (or 0 to 0), with the single repeat value, PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum*(1) set to 1000.

Governing quality metrics: Must be correct value.

See Also: PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.4.143 PD.IntTileMap.TruthTags.*TileNum*.Flg [L2,3,4,5]

Brief Definition: PD.IntTileMap.TruthTags.*TileNum*.Flg indicates the actual number of rows in the tiled data array, ES.Point.TruthTags.*TileNum*(ipt,j), when repetition factors are used in populating truth tag data in the tile indicated by the value of *TileNum*.

indices: none

Units/representation: Dimensionless, UInt16

Treatment/usage: Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.TruthTags.*TileNum*.Flg determines whether repetition factors are in use for ES.Point.TruthTags.*TileNum*(ipt,j) by separate value of *TileNum*. When it is greater than zero, it is the number of actual rows in the linearized indexing of the ipt-j data array when repetition factors are in use.

Governing quality metrics: Must have value >1 when repetition factor is in use for related items.

See Also: ES.Point.TruthTags.*TileNum*(ipt,j)

20.4.144 PD.IntTileMap.Tstart.*TileNum*.Flg [L1,2]

Brief Definition	PD.IntTileMap.Tstart.*TileNum*.Flg indicates the actual number of rows in the tiled data array, LD.WFClp.Tstart.*TileNum*(iray,ipi), when repetition factors are used for waveform clip start time data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.Tstart.*TileNum*.Flg determines whether repetition factors are in use for LD.WFClp.Tstart.*TileNum*(iray,ipi) by separate value of *TileNum*. When it is greater than zero, it is the number of actual rows in the linearized indexing of the ipi-iray data array when repetition factors are in use.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	LD.WFClp.Tstart.*TileNum*(iray,ipi)

20.4.145 PD.IntTileMap.TxCount.*TileNum*.Flg [L1,2,3,4,5]

Brief Definition	PD.IntTileMap.TxCount.*TileNum*.Flg indicates the actual number of rows in the tiled data array, SC.Pulse.TxCount.*Il.TileNum*(ip,j), when repetition factors are used for IPP duration data in the tile indicated by the value of *TileNum*.
indices	none
Units/representation	Dimensionless, UInt32
Treatment/usage	Per the discussion of 10.2.1.5 , when a PD.IntTileMap.*.*TileNum*.Flg is set or defaulted to the value, 0, the corresponding tile is an ordinary array and the actual "row" index equals the logical "row" index. The use of quotes around row is to signify that the fastest changing index (rightmost, due to CMMD row major convention) has a (degenerate) range of only one value. When a PD.IntTileMap.xxx.*TileNum*.Flg is set to a value greater than 0, the corresponding tile has a repetition factor for each consecutively repeated value of the quantity carried in the tile. In this case the fastest changing index

	has a value range of 2: for example from 1 to 2, where the first is the repetition factor, and the second is the repeated value. PD.IntTileMap.TxCount.*TileNum*.Flg determines whether repetition factors are in use for SC.Pulse.TxCount.*Il.TileNum*(ip,j) by separate value of *TileNum*. When it is greater than zero, it is the number of actual rows in the linearized indexing of the ip indexed data array when repetition factors are in use.
Governing quality metrics	Must have value >1 when repetition factor is in use for related items.
See Also	SC.Pulse.TxCount.*Il.TileNum*(ip,j)

PD.UserDefined.*

PD.UserDefined.* supplies defining information for user defined data items UD.* and PF.UserDefined.*

20.4.146 PD.UserDefined.NB_Def.P*n* [L1,2,3,4,5]

Brief Definition	PD.UserDefined.NB_Def.P*n* is the definition text for the user defined narrow band **p**arameter identified by P*n*.
indices	None. The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined narrow band parameter. So, in the case where a user has definitions for a first and second narrow band parameter, the data set would contain two definition strings. These might be, for example, PD.UserDefined.NB_Def.P1 and PD.UserDefined.NB_Def.P2.
Units/representation	Dimensionless, BCS Character string of length PD.Wordsize.NB_Def.P*n*
Treatment/usage	Guidelines for definition string content are provided in 10.2.3. Items defined here can be given a distinct value for each tile, but the definition is independent of the tile for which a value is provided. Thus, *TileNum* is not a part of the name of this parameter, but is part of the data tag being defined, PF.UserDefined.NB_Dat.P*n*.*TileNum*(j).
Governing quality metrics	Adherence to guidelines of 10.2.3.
See Also	PF.UserDefined.NB_Dat.P*n*.*TileNum*(j) PD.Wordsize.NB_Def.P*n* PD.UserDefined.PtDef.P*n*

20.4.147 PD.UserDefined.NB_Type.P*n* [L1,2,3,4,5]

Brief Definition	PD.UserDefined.NB_Type.P*n* is a code for the data type (representation) of user defined narrow band data item P*n*.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.UserDefined.NB_Type.P*n* complements PD.UserDefined.NB_Def.P*n* to fully describe a user defined data item. This item will be populated with the

appropriate code choice from the following table when a user defined narrow band data item PF.UserDefined.NB_Dat.*TileNum*(j) is being furnished.

Code	Meaning
1	UInt8
2	UInt16
3	UInt32
4	Int8
5	Int16
6	Int32
7	Single precision Float (real)
8	Double precision Float (real)
9	Single precision Float (complex, 64 bits total)
10	Character string

Governing quality metrics

See Also PF.UserDefined.NB_Dat.*TileNum*(j)

20.4.148 PD.UserDefined.PtDef.P*n*[L2,3,4,5]

Brief Definition PD.UserDefined.PtDef.P*n* is the definition text for the user defined wide band point oriented parameter identified by P*n*.

indices None.

The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined wide band point oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two definition strings. These might be, for example, PD.UserDefined.PtDef.P1 and PD.UserDefined.PtDef.P2.

Units/representation Dimensionless, Character string of length PD.Wordsize.PtDef.P*n*

Treatment/usage Guidelines for definition string content are provided in 10.2.3. Items defined here are wide band and, to support selective retrieval, are tiled. But the definitions do not vary from tile to tile. Thus, *TileNum* is not a part of the name of this parameter, but is part of the data tag being defined, UD.PtDat.P*n*.*TileNum*(ipt).

Governing quality metrics

See Also PD.Wordsize.PtDef UD.PtDat.P*n*.*TileNum*(ipt). PD.UserDefined.NB_Def.P*n*

20.4.149 PD.UserDefined.PtType.P*n*[L2,3,4,5]

Brief Definition: PD.UserDefined.PtType.P*n* is a code for the data type (representation) of user defined wide band point oriented data item P*n*.

indices: None

The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined wide band point oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two definition strings. These might be, for example, PD.UserDefined.PtType.P1 and PD.UserDefined.PtType.P2.

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.UserDefined.PtType.P*n* complements PD.UserDefined.PtDef.P*n* to fully describe a user defined data item. This item will be populated with the appropriate code choice from the following table when a user defined narrow band data item UD.PtDat.P*n*.*TileNum*(ipt) is being furnished.

Code	Meaning
1	UInt8
2	UInt16
3	UInt32
4	Int8
5	Int16
6	Int32
7	Single precision Float (real)
8	Double precision Float (real)
9	Single precision Float (complex, 64 bits total)
10	Character string

Governing quality metrics

See Also: UD.PtDat.P*n*.*TileNum*(ipt)

20.4.150 PD.UserDefined.PulseDef.P*n*[L1,2,3,4,5]

Brief Definition: PD.UserDefined.PulseDef.P*n* is the definition text for the user defined wide band pulse oriented parameter identified by P*n*.

indices: None.

The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined wide band pulse oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two definition strings. These might be, for example, PD.UserDefined.PulseDef.P1 and

PD.UserDefined.PulseDef.P2.

Units/representation: Dimensionless, BCS Character string of length PD.Wordsize.PulseDef.P*n*

Treatment/usage: Guidelines for definition string content are provided in 10.2.3. Items defined here are wide band and, to support selective retrieval, are tiled. But the definitions do not vary from tile to tile. Thus, *TileNum* is not a part of the name of this parameter, but is part of the data tag being defined, UD.PulseDat.P*n*.*TileNum*(ipt).

Governing quality metrics

See Also: PD.Wordsize.PulseDef UD.PulseDat.P*n*.*TileNum*(ipt).

20.4.151 PD.UserDefined.PulseType.P*n*[L1,2,3,4,5]

Brief Definition: PD.UserDefined.PulseType.P*n* is a code for the data type (representation) of user defined wide band pulse oriented data item P*n*.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: PD.UserDefined.PulseType.P*n* complements PD.UserDefined.PulseDef.P*n* to fully describe a user defined data item. This item will be populated with the appropriate code choice from the following table when a user defined narrow band data item UD.PulseDat.P*n*.*TileNum*(ip) is being furnished.

Code	Meaning
1	UInt8
2	UInt16
3	UInt32
4	Int8
5	Int16
6	Int32
7	Single precision Float (real)
8	Double precision Float (real)
9	Single precision Float (complex, 64 bits total)
10	Character string

Governing quality metrics

See Also: UD.PulseDat.P*n*.*TileNum*(ip)

PD.WFClp.*

PD.WFClp.* supplies sizing information for waveform clip data

20.4.152 PD.WFClp.Nsamps.*TileNum*(iray/irays, ipi/j) [L2]

Brief Definition

PD.WFClp.Nsamps.*TileNum*(iray/irays, ipi/j) gives the number of samples in each waveform clip (clips are identified by a pulse and ray number) in a tile which is identified by *TileNum* (*TileNum* is specific to a return number).

indices

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile".

Use of *TileNum* for wave form clips enterprise L-2:

Waveforms in enterprise L-2 data are subsets ("clips") of the full receive window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "1R*nnnnnn*" needs to be unique.

Default case: { PD.IntTileMap.Nsamps.*TileNum*.Flg unpopulated or explicitly set to 0}

Index, iray, ranges from PD.TileMap.RayRange(j_tile, 1) to PD.TileMap.RayRange(j_tile, 2), that is from the first ray in the tile to the last ray in the tile. The tile association of j_tile appearing in the ray range data and sub-string *TileNum* in PD.WFClp.Nsamps.*TileNum*(iray, ip) is made via PD.TileMap.TileNum(j_tile). The definitions of iray, if populated, will either be present in SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) or SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). The *TileNum* specific ray descriptions create a unique shortened list of rays with consecutive values of "iray" for each specific tile. When these are in use, "iray" runs over only the rays that have populated waveform clips in the wideband data. While enterprise level-1 data sets will tend to use the global iray definitions, enterprise level-2 data sets will vary depending on the level-2 generating application and the determination of whether there are few enough waveforms to warrant using a small, tile oriented, waveform list approach to minimize data volume.

Index, ipi, ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*.

Repetition factor in use case: { PD.IntTileMap.Nsamps.*TileNum*.Flg explicitly set to value greater than 0} See internal tile mapping.
The index, iray, is replaced by a dummy row index ("irays") that ranges from 1 to PD.IntTileMap.Nsamps.*TileNum*.Flg, that is over the number of listed iray entries in the repetition version of the array.

The pulse index, "ip", is replaced by a new dummy index, j, =1,2,3. This index supports repetition factor use to reduce data volume. Values of PD.WFClp.Nsamps.*TileNum*(irays,1) are iray numbers and PD.WFClp.Nsamps.*TileNum*(irays,2) are repetition factors and PD.WFClp.Nsamps.*TileNum*(irays,3) will have Nsamps values.
Care must be taken by the data set writing software to assure that the values of ipi and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.

Units/representation: Dimensionless, UInt32

Treatment/usage: Default case:
PD.WFClp.Nsamps.*TileNum*(iray, ipi) is the number of fast time samples in the waveform clips corresponding to the return number encoded in *TileNum*, and the value of the ray and pulse associated with the row and column indices of a matrix whose elements are $\text{Nsamps}_{\text{iray,ipi}}$.
PD.WFClp.Nsamps.*TileNum*(iray, ipi) thus gives the useful range of index "it" in LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) on an individual data vector basis. LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) is designed to be able to hold received waveform clips of varying fast time durations. The upper bound for "it" in the array is sized for the waveform clip with the longest data vector in the array for *TileNum*.

Repetition factor use case:
PD.WFClp.Nsamps.*TileNum*(irays,2) is the number of times the value in PD.WFClp.Nsamps.*TileNum*(irays,3) consecutively repeats in single stretch of "ipi" values corresponding to no change over a range of pulses. In other words, PD.WFClp.Nsamps.*TileNum*(irays,2) is the number of consecutive 'used' pulses (indexed by ipi) for a ray given by PD.WFClp.Nsamps.*TileNum*(irays,1) for which the value of Nsamps is given

in PD.WFClp.Nsamps.*TileNum*(irays,3).

PD.WFClp.Nsamps.*TileNum*(irays,3) is the number of fast time samples (Nsamps) in the waveform clip described for ray index, PD.WFClp.Nsamps.*TileNum*(irays,1), for the pulse number which can be recovered by accumulating the repeat factors in PD.WFClp.Nsamps.*TileNum*(irays,2) per the discussion in 10.2.1.5.

Any pulse-ray value not covered in the reduced 'Nsamps' table [PD.IntTileMap.Nsamps.*TileNum*.Flg rows of 3 numbers] is defaulted to Nsamps=0. See internal tile mapping.

For illustration, consider the table below.

irays	j=1	j=2	j=3
1	iray=1	100	Nsamps=1024
2	iray=1	200	Nsamps=512
3	iray=1	100	Nsamps=768
4	iray=1	1000	Nsamps=1024
5	iray=2	1400	Nsamps=256
6	iray=3	100	Nsamps=1024
7	iray=3	200	Nsamps=512
8	iray=3	100	Nsamps=64

This set of 24 numbers replaces a 3 by 1400 array of numbers. We have a compression factor of 525. This says that:

- PD.IntTileMap.Nsamps.*TileNum*.Flg=8
- the equivalent default "uncompressed" array PD.WFClp.Nsamps.*TileNum*(iray, ip) describes 3 rays and 1400 pulses and had

PD.WFClp.Nsamps.*TileNum*(1, 1 through 100) =1024
PD.WFClp.Nsamps.*TileNum*(1, 101 through 300) =512
PD.WFClp.Nsamps.*TileNum*(1, 301 through 400) =768
PD.WFClp.Nsamps.*TileNum*(1, 401 through 1400)=1024
PD.WFClp.Nsamps.*TileNum*(2, 1 through 1400)=256
PD.WFClp.Nsamps.*TileNum*(3, 1 through 100) =1025
PD.WFClp.Nsamps.*TileNum*(3, 101 through 300) =512
PD.WFClp.Nsamps.*TileNum*(3, 301 through 400) =64
PD.WFClp.Nsamps.*TileNum*(3, 401 through 1400) =0

The first 8 statements above come from accumulating the repeat factors[199] in PD.WFClp.Nsamps.*TileNum*(irays, j). The last line follows from the default rule that items not defined in the table have value zero. It should be noted that the 'unwinding' of the pulse numbers starts over for each ray. Note that in the first 8 lines, there were no Nsamps=0 cases. This is because we only use pulses that have some returns on at least one ray. That is the point of the "ipi" indexing for pulses that are a selection of only active values of local IPP numbers managed via PD.WFClp.PulseNum.*TileNum*(ipi). For a ray to have no clips associated for any pulses, it would have to appear in the above table only once (a trivial unwinding case). The repeat number (j=2) would be the number of pulses in the tile and the indicator for that would be Nsamps (j=3) having a value of 0. This could arise if the rays were globally defined and the rays with no clips happened to sort into the ray list before the last ray that does have a clip return reported. The more likely case is that we use tile specific ray definitions and those will be restricted to only rays that have clips in the tile. Taken with the use of only pulses that have clips (i.e., ipi indexing summarized immediately below), there will be no Nsamps=0 entries in such a table.

The tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter ip starts over from 0 for each tile. The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts, 3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent, PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases).

Care must be taken by the data set writing software to assure that the values of ip and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a

[199] We shall call this process "unwinding" the pulse numbers

specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.

Governing quality metrics: Must have correct data

See Also: PD.TileMap.TileNum(j_tile) PD.IntTileMap.Nsamps.*TileNum*.Flg PD.WFClp.PulseNum.*TileNum*(ipi) ES.WFClp.RtnsCode.*TileNum*(iray, ipi) LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) SC.WF.Rx.Nsamps(ich)

20.4.153 PD.WFClp.PulseNum.*TileNum*(ipi) [L2]

Brief Definition: PD.WFClp.PulseNum.*TileNum*(ipi) assigns an index, ipi, to each entry in the list of "used pulses" with populated waveform clips in a tile by *TileNum*, which is specific to a return number.

indices: Index, ipi, ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile".

Use of *TileNum* for wave form clips enterprise L-2:

Waveforms in enterprise L-2 data are subsets ("clips") of the full receive window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "1R*nnnnnn*" needs to be unique.

The *TileNum* values appearing in PD.WFClp.PulseNum.*TileNum*(ipi) must match the values in associated tiles, LD.WFClp.RxDatVec.*TileNum*(ip,iray,it).

Units/representation: Dimensionless, UInt16

Treatment/usage: PD.WFClp.PulseNum.*TileNum*(ipi) acts as a table that assigns a sparse list of pulse numbers to a wave clip tile specific index consecutive sequence "ipi" so that we do not need to provide large matrices with lots of zeros for pulses that don't have clips provided at level-2. The large matrix with lots of zeros would result from using the data set global pulse index, ip, rather than

focusing on a tile local set of wave clips from a limited number of not necessarily consecutive pulses. It is highly recommended that the ipi index be the ordinal values of the sparse list of ip values in a specific tile. The index, "ipi", replaces "ip" in many fields at L-2 so that sparse pulse wave form population is possible. At L-1, every pulse wave form (RxDatVec) must be in the file so an "ip" that just counts IPPs will not waste storage. But only a small subset of pulses may have waveforms at L-2, so ipi is not a consecutive counter of IPPs, only of "used IPPs.

The primary application for PD.WFClp.PulseNum.*TileNum*(ipi) is to permit effective management of the data array, ES.WFClp.Locs.*TileNum*(ipi,iray,j). However, the shortened consecutive pulse index list may be exploited in other enterprise level-2 versions of data with a pulse index.
The table below illustrates a simple case. For the fifth first return wave form clip tile (illustration uses *TileNum*= "1R005") the following table describes the 5 wave form clips out of the ten thousand pulses that could have been chosen to provide waveforms.

ipi—consecutive index value for a wave clip tile	**ip – the actual (local to a tile) pulse number a clip came from**
1	PD.WFClp.PulseNum.1R005(1)=2007
2	PD.WFClp.PulseNum.1R005(2)=2403
3	PD.WFClp.PulseNum.1R005(3)=5017
4	PD.WFClp.PulseNum.1R005(4)=6057
5	PD.WFClp.PulseNum.1R005(5)=7897

Put in the context of the larger CMMD enterprise level-2 waveform pulse labeling strategy, there are three elements:

- the tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter, ip, starts over from 0 for each tile. Note that SC.Pulse.SensorPos.*TileNum*(ip,j) provides sensor position in tile level pulse counts.
- Translation to actual IPP (that's the pulse number since start of collected data): The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3

) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases)

- Channel identification for a given wave form is in terms of pulse and ray via ES.WFClp.ChNum.*TileNum*(ipi,iray) which for a given waveform data vector from a known tile does not require conversion of ipi to ip support channel look up.

Care must be taken by the data set writing software to assure that the values of ip and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.

Governing quality metrics

See Also ES.WFClp.Locs.*TileNum*(ip**i**,iray,j) PD.HiIndx.WFClpPulseNum.*TileNum* ES.WFClp.ChNum.*TileNum*(ipi,iray)

PD.Wordsize.*

PD.wordsize.* supplies the wordsize for those items whose representation is given as variable precision in this document. This small set of items is needed so that the metadata can be transport independent. Not all transports provide a service for tracking the wordsize of data parameters.

20.4.154 PD.Wordsize.ChCombFunc [L2,L3]

Brief Definition PD.Wordsize.ChCombFunc provides the number of characters in the PF.Channlzn.Hist.Comb.Func(k) strings.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage PF.Channlzn.Hist.Comb.Func(k) is a list of characters strings, one for each channel of data in the current data set. They are all the same length and the number of characters is PD.Wordsize.ChCombFunc.

Governing quality metrics

See Also PF.Channlzn.Hist.Comb.Func(k)

20.4.155 PD.Wordsize.InfAssur [L1,2,3,4,5]

Brief Definition PD.Wordsize.InfAssur documents the number of bytes in the MD.InfAssur.* bit strings.

indices none

Units/representation Dimensionless, UInt16

Treatment/usage The number of bytes used in MD.InfAssur.NonRepud and MD.InfAssur.Seal is made variable up to very long strings to support a variety of cryptographic strengths.

Governing quality metrics

See Also MD.InfAssur.Seal

20.4.156 PD.Wordsize.Iray [L1,2]

Brief Definition PD.Wordsize.Iray documents the number of bytes in the ray identifier for wide band points and waveform clips.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage The number of bytes used in combining the source focal plane element identifier and array index into a ray index for a lidar or synthetic lidar point is permitted to be 1, 2 , 4, or 8. The data type notation is in terms of bits and the modern choice of 8 bits per byte is presumed.

The following table indicates the wordsize for PF.Point.IrayNum.*TileNum*(ipt,j):

Value of PD.Wordsize.Iray	**Data Type for PF.Point.IrayNum.*TileNum*(ipt,j)**
1	UInt8
2	UInt16
4	UInt32
8	Int64

Governing quality metrics

See Also PF.Point.IrayNum.*TileNum*(ipt,j)

20.4.157 PD.Wordsize.Intnsy [L2,3,4,5]

Brief Definition: PD.Wordsize.Intnsy documents the number of bytes in the LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) or LD.Point.Intensity.*TileNum*(ipt,j) arrays.

indices: none

Units/representation: Dimensionless, UInt8

Treatment/usage: The number of bytes used in combining the source focal plane element identifier and array index into a ray index for a lidar or synthetic lidar point is permitted to be 1, 2 , 4, or 8. The data type notation is in terms of bits and the modern choice of 8 bits per byte is presumed.

The following table indicates the wordsize for LD.Point.Intensity.*:

Value of PD.Wordsize.Intnsy	**Data Type for LD.Point.Intensity.***
1	UInt8
2	UInt16
4	UInt32
8	Int64

Governing quality metrics

See Also: LD.Point.Intensity.*ChNum*.*TileNum*

20.4.158 PD.Wordsize.Meta(j) [L1,2,3,4,5]

Brief Definition: PD.Wordsize.Meta(j) provides the number of characters in several metadata character stings

indices: Index j ranges from PD.LoIndx.MD_Strings to PD.HiIndx.MD_Strings, that is, over the metadata items with somewhat long and variable length character strings.

Units/representation: Dimensionless, UInt16

Treatment/usage: This metadata item is a convenience feature to support tools that will put some of the character string metadata onto GUIs for display. This will make it much simpler to control wasted space and assure adequate space to present information that is mainly textual, and would be largely for informing human readers. A good example is MD.Objective.Abstract.

j	**String whose length in characters is given**
1	MD.Objective.Abstract
2	MD.Objective.Desc
3	MD.Objective.Tasker

Governing quality metrics

See Also MD.Objective.Abstract PD.HiIndx.MD_Strings

20.4.159 PD.Wordsize.Pos [L2,3,4,5]

Brief Definition PD.Wordsize.Pos documents the number of bytes in the components of wide band point coordinates.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage The number of bytes used in each of the components (e.g., X,Y, and Z) of a lidar or synthetic lidar point is permitted to be 1, 2 , 4, or 8. The bit depths of each component are equal. This permits handling the data as a vector without extra overhead and costs less than 20% worst case in data set size. It also keeps all values on addressable boundaries and avoids the need to unpack bytes to read values.

The following table indicates the wordsize for LD.Point.Pos.*ChNum*.*TileNum*(ipt,j):

Value of PD.Wordsize.Pos	**Data Type for LD.Point.Pos.*ChNum*.*TileNum*(ipt,j)**
1	UInt8
2	UInt16
4	UInt32
8	Int64

Note: The unsigned integers do not force a single octant for point data. This is because the stored data are run through a linear transformation to convert to positions in meters. While the scale factor for all points in a tile is a single sign, the XYZ offsets from the tile origin can be positive or negative. So we always have a choice about the number of octants that the data occupies in a tile.

Governing quality metrics

See Also LD.Point.Pos.*ChNum*.*TileNum*(ipt,j)

20.4.160 PD.Wordsize.SensorPos [L1,2,3,4,5]

Brief Definition PD.Wordsize.SensorPos documents the number of bytes in the stored integer X,Y, and Z elements of the pulse indexed sensor position data.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage The number of bytes used in each of the X,Y, and Z components of a lidar sensor position is allowed to be 1, 2 , 4, or 8. The bit depths of X,Y and Z are equal. This permits handling the data as a vector without extra overhead and costs less than 20% worst case in data set size. It also keeps all values on addressable boundaries and avoids the need to unpack Bytes to read values. The following table indicates the codes that may be used to populate the wordsize for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j):

Value of PD.Wordsize.SensorPos	**Data Type for** SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j)
1	Int8
2	Int16
4	Int32
8	Int64

Governing quality metrics

See Also SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j)

20.4.161 PD.Wordsize.Pulse [L1,2,3,4,5]

Brief Definition PD.Wordsize.Pulse documents the number of bytes in ES.Point.PulseNum.*TileNum*(ipt, j).

indices none

Units/representation Dimensionless, UInt8

Treatment/usage The number of bytes used in ES.Point.PulseNum.*TileNum*(ipt, j) is 1, 2 , 4, or 8. This permits Reducing the wordsize in large data sets by using a pulse count offset for each tile of point data, PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k). It also keeps all values on addressable boundaries and avoids the need to unpack Bytes to read values. The following table indicates the wordsize for ES.Point.PulseNum.TileNum(ipt, j):

Value of PD.Wordsize.Pulse	**Data Type for ES.Point.PulseNum.TileNum(ipt, j)**
1	UInt8
2	UInt16
4	UInt32
8	Int64

Governing quality metrics

See Also PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k)
ES.Point.PulseNum.*TileNum*(ipt, j)

20.4.162 PD.Wordsize.Recon [L1,2,3,4,5]

Brief Definition	PD.Wordsize.Recon documents the number of characters in processing algorithm input data lists
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	PD.Wordsize.Recon is the number of text characters in an element of the PF.Recon.CommandFiles(j) data vector. This will be set to a value large enough to represent the largest comma delimited name-value input parameter list for any of the executed applications listed in PF.Recon.App(j).
Governing quality metrics	
See Also	PF.Recon.CommandFiles(j) PF.Recon.App(j).

20.4.163 PD.Wordsize.RxDatVec [L1,2]

Brief Definition	PD.Wordsize.RxDatVec provides the number of bits per sample of data in a received waveform fast time data vector.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	PD.Wordsize.RxDatVec determines the number of bits per sample in every LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) fast time receive waveform vector. Typically, the value of PD.Wordsize.RxDatVec would be 8 or 16 bits for linear mode data. For Geiger mode data, the value represents a fast time index, and may be a large integer value, so the number of bits indicated by PD.Wordsize.RxDatVec is possibly as high as 32. When pulse compression is used, a vector quantization decoding may be used. If this is the case, SC.WF.VQ_Decode(ia,icode) will be populated. Also if VQ codes are in the fast time data vector, they may be non-integer multiples of 8 bits to support bandwidth compression. In that case, PD.Wordsize.RxDatVec might be a value like 10 bits. Those 10 bits would be loaded into the lower bits of an integer that is used as the look up index in the decoding table. If a sensor stores intensities in samples that are not multiples of 8 bits, the use of the correct value of PD.Wordsize.RxDatVec supports retaining the wide band data in non-standard integer words until arithmetic operations need to be performed. The bit offset of the nth sample can be computed and the correct bit sequence can be extracted via S/W processes or H/W processes when available. So, to support multiple scenarios where the wide band data might not be stored in power of 2 byte counts, the integer supplied as a value of PD.Wordsize.RxDatVec is always directly interpreted as a number of bits, not bytes.

	PD.Wordsize.RxDatVec exists so that the individual values of fast time samples can be read out of the I/O record conveniently. The first bit of the LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) vector for a given value of ipi and iray, is the first bit of the first sample, and so on through the data vector.
Governing quality metrics	Must be the correct value.
See Also	LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) SC.WF.VQ_Decode(ia,icode) MD.Sensor.lidarMode

20.4.164 PD.Wordsize.TileNum [L1,2,3,4,5]

Brief Definition	PD.Wordsize.TileNum gives number of characters in the evaluated *TileNum* substrings used in the data set.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	Per the discussion of 10.2.1.2, tiled parameters in the lidar CMMD have a computed sub-string to indicate specific instantiations of data items for which there may be many instantiations, one for each vertical tile of data. PD.Wordsize.TileNum gives the number of characters in such a string so that we can tailor the size of the tagnames to the data set. The value of PD.Wordsize.TileNum should generally not exceed 10.
Governing quality metrics	
See Also	e.g. LD.Point.Pos.*ChNum*.*TileNum*(ipt,j), LD.Point.Intensity.*TileNum*(ipt, j)

20.4.165 PD.Wordsize.NB_Def.P*n*[L1,2,3,4,5]

Brief Definition	PD.Wordsize.NB_Def.P*n* gives number of characters in the user definition supporting user defined narrow band items in the data set.
indices	None The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined narrow band parameter. So, in the case where a user has definitions for a first and second narrow band parameter, the data set would contain two definition strings. The number of characters for each would be placed in, for example, PD.Wordsize.NB_Def.P1 and PD.Wordsize.NB_Def.P2.
Units/representation	Dimensionless, UInt16
Treatment/usage	Per 10.2.3, this CMMD supports user defined narrow band data via the trio of PF.UserDefined.NB_Dat.P*n*.*TileNum*(j), PD.UserDefined.NB_Def.P*n*, and PD.UserDefined.NB_Type.P*n*. The definition text string for each value of PD.UserDefined.NB_Def.P*n* is provided as a separate data item and the number of characters in each is given by the value stored with the corresponding tag, PD.Wordsize.NB_Def.P*n*. Thus, PD.Wordsize.NB_Def.P1

is the number of characters in the text string, PD.UserDefined.NB_Def.P1.

Governing quality metrics

See Also PD.UserDefined.NB_Def.P*n* PF.UserDefined.NB_Dat.P*n*.*TileNum*(j)

20.4.166 PD.Wordsize.PtDef.P*n* [L2,3,4,5]

Brief Definition PD.Wordsize.PtDef.P*n* gives number of characters in the user definition supporting user defined wide band point oriented items in the data set.

indices None

The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined wide band point oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two definition strings. The number of characters for each would be placed in, for example, PD.Wordsize.PtDef.P1 and PD.Wordsize.PtDef.P2.

Units/representation Dimensionless, UInt16

Treatment/usage Per 10.2.3, this CMMD supports user defined wide band data via the trio of UD.UserDefined.PtDat.Pn.*TileNum*(ipt), PD.UserDefined.PtDef.P*n*, and PD.UserDefined.PtType.P*n*. The definition text string for each value of PD.UserDefined.PtDef.P*n* is provided as a separate data item and the number of characters in each is given by the value stored with the corresponding tag, PD.Wordsize.PtDef.P*n*. Thus, PD.Wordsize.PtDef.P1 is the number of characters in the text string, PD.UserDefined.PtDef.P1. .

Governing quality metrics

See Also PD.UserDefined.PtDef.P*n*

20.4.167 PD.Wordsize.PulseDef.P*n* [L1,2,3,4,5]

Brief Definition PD.Wordsize.PulseDef.P*n* gives number of characters in the user definition supporting user defined wide band pulse oriented items in the data set.

indices None

The evaluated sub-string "*n*" behaves somewhat as an index, however. A separate value of *n* is used for each user defined wide band pulse oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two definition strings. The number of characters for each would be placed in, for example, PD.Wordsize.PulseDef.P1 and PD.Wordsize.PulseDef.P2.

Units/representation Dimensionless, UInt16

Treatment/usage Per 10.2.3, this CMMD supports user defined wide band data via the trio of UD.UserDefined.PulseDat.P*n*.*TileNum*(ipt), PD.UserDefined.PulseDef.P*n*, and PD.UserDefined.PulseType.P*n*. The definition text string for each value of PD.UserDefined.PulseDef.P*n* is provided as a separate data item and the

	number of characters in each is given by the value stored with the corresponding tag, PD.Wordsize.PulseDef.P*n*. Thus, PD.Wordsize.PulseDef.P1 is the number of characters in the text string, PD.UserDefined.PulseDef.P1. .
Governing quality metrics	
See Also	PD.UserDefined.PulseDef.P*n*

20.4.168 PD.Wordsize.Truth(j) [L1,2,3,4,5]

Brief Definition	PD.Wordsize.Truth(j) gives number of characters in the kmz truth illustration strings.
indices	Index, j, ranges from PD.LoIndx.Truth(2) to PD.HiIndx.Truth(2), that is over the range of truth illustrations.
Units/representation	Dimensionless, UInt16
Treatment/usage	This determines the size of the zipped KML text strings that supply illustrations of the neighborhoods of ground truth points in ES.Truth.IllusKMZ.*j*. So the number of characters, nchar ,in the zipped string, ES.Truth.IllusKMZ.*j* is given by PD.Wordsize.Truth(j). It is advised that the evaluated string, *j*, be kept in visual correspondence with the value of the corresponding j, the index in PD.Wordsize.Truth(j). For example, the size of the character string, ES.Truth.IllusKMZ.2 is the value of PD.Wordsize.Truth(2). Thus, PD.Wordsize.Truth(j) is the number of bytes in the field, ES.Truth.IllusKMZ.*j*.
Governing quality metrics	
See Also	PD.HiIndx.Truth(2) ES.Truth.IllusKMZ.*j*

20.4.169 PD.Wordsize.TxDat [L1,2,3]

Brief Definition	PD.Wordsize.TxDat gives the number of bits per sample of SC.WF.TxProfile.Dat(il,j).
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	The transmit wave form data will either capture intensity as a function of fast time, or it will capture VQ encoded quadrature sampled mixed down[200] waveform data. PD.Wordsize.TxDat determines the number of bits per sample in the SC.WF.TxProfile.Dat(il,j) fast time sampled transmit waveform vector. Typically, the value of PD.Wordsize.TxDat would be 8 or 16 bits for linear

[200] This includes a non-shifted frequency spectrum.

and Geiger mode data.
This contrasts with PD.Wordsize.RxDatVec for Geiger mode data, where the value represents a fast time index, and may be a large integer value, so the number of bits indicated by PD.Wordsize.RxDatVec is possibly as high as 32. In Geiger mode, we still need to understand the temporal structure of the pulse intensity envelope, and there is no particular time of interest analogous to the photon detection time on the receive side.

When pulse compression is used, a vector quantization decoding may be used. If this is the case, SC.WF.VQ_Decode(ia,icode) will be populated. Also, if VQ codes are in the fast time data vector, they may be non-integer multiples of 8 bits to support bandwidth compression. In that case, PD.Wordsize.TxDat might be a value like 10 bits. Those 10 bits would be loaded into the lower bits of an integer that is used as the look up index in the decoding table.
If a sensor stores intensities in samples that are not multiples of 8 bits, the use of the correct value of PD.Wordsize.TxDat supports retaining the SC.WF.TxProfile.Dat(il,j) data in non-standard integer words until arithmetic operations need to be performed. The bit offset of the nth sample can be computed and the correct bit sequence can be extracted via S/W processes or H/W processes when available. So, to support multiple scenarios where the wide band data might not be stored in power of 2 byte counts, the integer supplied as a value of PD.Wordsize.TxDat is always directly interpreted as a number of bits, not bytes.
PD.Wordsize.TxDat exists so that the individual values of fast time samples can be read out of the I/O record conveniently. The first bit of row "il" of SC.WF.TxProfile.Dat(il,j) is the first bit of the first sample of a word that is PD.Wordsize.TxDat bits long. The PD.Wordsize.TxDat+1 bit is the first bit of the second sample, and so on through the data vector.

Governing quality metrics

See Also SC.WF.TxProfile.Dat(il,j)

20.4.170 PD.Wordsize.Zones [L1,2,3,4,5]

Brief Definition PD.Wordsize.Zones gives number of characters in the MGRS zone designators used in the current data set

indices none

Units/representation Dimensionless, UInt8

Treatment/usage

This determines the number of characters in the strings of the data vector MD.Coverage.Bounds.ZoneList(j). This drives the spatial detail with which the zones are used in the data set. If MD.Coverage.Bounds.ZoneList(j) is populated, PD.Wordsize.Zones will generally take a value from the following table:

Value of PD.Wordsize.Zones	Size of zones being referenced
7	10km
9	1km
11	100m
13	10m

Governing quality metrics

See Also

MD.Coverage.Bounds.ZoneList(j)

20.5 SC.* Sensor Configuration

Sensor Configuration data describe the sensor with regard to its mode of operation and other settings, the size and shape of the focal plane photon detection regions, spectral and directional sensitivity, scan and subswathing strategies and other hardware and operational parameters as described below.

SC.Array.*

SC.Array.* parameters describe the focal plane detection grid physically. The term array is used in the generic sense of an electro-magnetically and spatially diverse set of optically sensitive elements. The "array" could be a single element, a line or a matrix of individual elements.

20.5.1 SC.Array.BadElmnts.*Ia*(j) [L1,2]

Brief Definition	SC.Array.BadElmnts.*Ia*(j) indicates which elements in each array should be skipped over in processing due to unusable data.
indices	Substring, "*Ia*", acts as an index and the character(s) represents integers in the range from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays. This is an evaluated substring so that the length of the lists of bad elements can be adjusted to the needs of each array. Index, j , ranges from PD.LoIndx.Array.Elem(ia,2) to PD.HiIndx.Array.Elem(ia,2), that is, over the bad elements in array ia. The index "ia" here is the integer determined by the value of the substring, "*Ia*". For example, if the substring,"*Ia*", is populated with the characters, "02", then the corresponding value of ia is 2.
Units/representation	Dimensionless, UInt32
Treatment/usage	The processing of enterprise level-1 to level-2 data should exclude passing data onto point cloud or waveform clips if it is due to the array/element values indicated by SC.Array.BadElmnts.*Ia*(j). Any channel using an array indicated in *Ia* will be affected by the corresponding SC.Array.BadElmnts.*Ia*(j). SC.Array.BadElmnts.*Ia*(j) is a list of "ie" values for each array, *Ia*, in the sensor with enough bad elements to warrant a warning to manage time or quality issues. The assignment of a value of "ie" here must be consistent with the "ie" assignments used other data items such as SC.Array.RayDirs.*Ia*(icase,ie,j) and SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k).
Governing quality metrics	Proper correspondence with FilterDat
See Also	PD.LoIndx.Array.Elem(ia,j) SC.Array.RayDirs.*Ia*(icase,ie,j)

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.5.2 SC.Array.CrossTalkQ1.*Ia*(j,k,n) [L1,2]

Brief Definition

SC.Array.CrossTalkQ1.*Ia*(j,k,n) provides cross talk levels and delays for the lidar's detector array.

indices

Substring, "*Ia*", acts as an index and the character(s) represents integers in the range from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays. This is an evaluated substring so that the length of the cross talk data arrays can be adjusted to the needs of each array.

Index, j , ranges from PD.LoIndx.Array.Elem(ia,3) to PD.HiIndx.Array.Elem(ia,3), that is, over the rows in the cross talk data for array ia. The index "ia" here is the integer determined by the value of the substring, "*Ia*". For example, if the substring,"*Ia*", is populated with the characters, "02", then the corresponding value of ia is 2.

Index, k , ranges from PD.LoIndx.Array.Elem(ia,4) to PD.HiIndx.Array.Elem(ia,4), that is, over the columns in the cross talk data for array ia. The index "ia" is described above.

Index, n=1,2

Units/representation

SC.Array.CrossTalkQ1.*Ia*(j,k,n) is single float

SC.Array.CrossTalkQ1.*Ia*(j,k,1 (dimensionless)

SC.Array.CrossTalkQ1.*Ia*(j,k,2) (seconds)

Treatment/usage

Cross-talk is a focal plane detector array artifact that is the consequence of optical photons produced during the amplification (avalanche) process reaching other pixels and triggering further avalanches.

Consider a focal plane array consisting of rows of detecting elements. The minimum number of rows is one and the minimum number of elements per row is one. We index the rows with an index, say *i*, and the elements in each row with an index, say *j*. Regardless of whether the physical elements are arranged in columns on the focal plane, or some sort of (e.g. triangular) lattice, we can make a table of the detector elements where the rows of the table correspond to the rows of detectors and the columns of the table correspond to the element indices within the rows.

SC.Array.CrossTalkQ1.*Ia*(j,k,n) is a special case of such a table. For each of the detector elements, specified by a value of the row index and the value of the column index in SC.Array.CrossTalkQ1.*Ia*(j,k,n), we supply a coupling level and a time delay. What is special is that while we index by row and element, the first row and first element in SC.Array.CrossTalkQ1.*Ia*(j,k,n) is understood to apply to any particular detector in the focal plane array and the

values of j and k in SC.Array.CrossTalkQ1.*Ia* are actually row and element *offsets* from the focal plane element that is being assumed to be at the location of the first row and column of SC.Array.CrossTalkQ1.*Ia*(j,k,n). The "Q1" in this data tag refers to the first quadrant of a 2-D Cartesian frame on a plane.

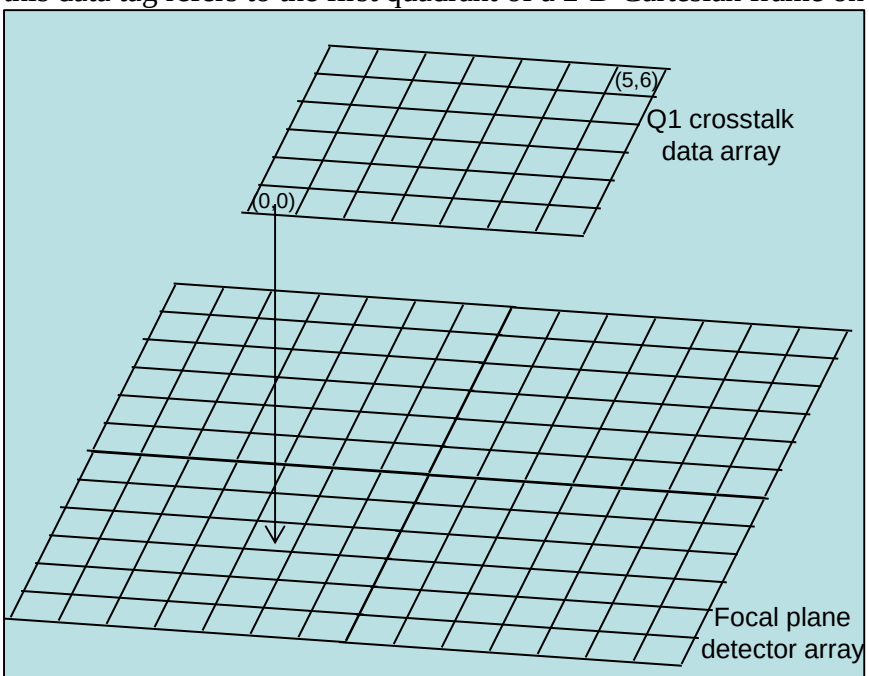


Illustration of a 6 row, 7 column crosstalk data grid with its origin "pixel" applied to an arbitrary detector on the focal plane.

For the sake of the following discussion, let us assume that PD.LoIndx.Array.Elem(ia,3) and PD.LoIndx.Array.Elem(ia,4) are both 0. Any elements of SC.Array.CrossTalkQ1.*Ia*(j,k,n) that would "fall off" the focal plane due to the choice of element to which the cross talk data grid origin is applied should be ignored in such a case.

SC.Array.CrossTalkQ1.*Ia*(j,k,1) is the ratio of the the response in the detector element referenced by (j,k) when no photon (flux) is incident on it to the response in the element referenced by (0,0) when a unit of photon flux is incident on that element. We are treating the *pattern* of response as homogeneous across the focal plane by supplying only one SC.Array.CrossTalkQ1.*Ia*(j,k,1) for an array. The variability of response across the array is captured approximately via an element dependent scaling factor that multiplies the whole SC.Array.CrossTalkQ1.*Ia*(j,k,1) data grid by a spatial constant (for a single pulse, ip) that is mapped to the actual element presumed to be at (0,0) in the SC.Array.CrossTalkQ1.*Ia*(j,k,1) data array.

That constant is found in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,2) Since no element can both have photon flux incident on it and not have photon flux incident on it, we require that SC.Array.CrossTalkQ1.*Ia*(0,0,1)= 0.000000E+00.

For linear mode family operation, the response ratio carried in SC.Array.CrossTalkQ1.*Ia*(j,k,1) is the fraction of the photo current at the detector element corresponding to (j,k)=(0,0) that becomes erroneously added to the output of the element offset by j rows and k elements.

For Geiger mode, SC.Array.CrossTalkQ1.*Ia*(j,k,1) is the probability of a response being triggered at offset (j,k) due only to the triggering of the element corresponding to (j,k)=(0,0).

If only SC.Array.CrossTalkQ1.*Ia* is populated, the values in SC.Array.CrossTalkQ2.*Ia* , SC.Array.CrossTalkQ3.*Ia*, SC.Array.CrossTalkQ4.*Ia* are implied by reflection of the data grid about the adjoining axis or through the origin.

If SC.Array.CrossTalkQ1 and SC.Array.CrossTalkQ2 or SC.Array.CrossTalkQ3 are populated, the other two quadrants are implied by reflection.
The vendor can provide all 4 quadrants if necessitated by substantially different patterns in each quadrant.

SC.Array.CrossTalkQ1.*Ia*(j,k,2) is the delay in seconds of the response in the detector element referenced by (j,k) when no photon (flux) is incident after the response in the element referenced by (0,0) when a unit of photon flux is incident on that element. We are treating the *pattern* of time delay as homogeneous across the focal plane by supplying only one SC.Array.CrossTalkQ1.*Ia*(j,k,2) for an array. It is assumed that the delays in cross talk response are independent of slow time. Since no element can both have photon flux incident on it and not have photon flux incident on it, we require that SC.Array.CrossTalkQ1.*Ia*(0,0,2)= 0.000000E+00.

Governing quality metrics

See Also

PD.HiIndx.Array.Elem(ia,3)
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.5.3 SC.Array.CrossTalkQ2.*Ia*(j,k,n) [L1,2]

Brief Definition

SC.Array.CrossTalkQ2.*Ia*(j,k,n) provides cross talk levels and delays for the

	lidar's detector array.
indices	Substring, "*Ia*", acts as an index and the character(s) represents integers in the range from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays. This is an evaluated substring so that the length of the cross talk data arrays can be adjusted to the needs of each array. Index, j , ranges from PD.LoIndx.Array.Elem(ia,3) to PD.HiIndx.Array.Elem(ia,3), that is, over the rows in the cross talk data for array ia. The index "ia" here is the integer determined by the value of the substring, "*Ia*". For example, if the substring,"*Ia*", is populated with the characters, "02", then the corresponding value of ia is 2. Index, k , ranges from PD.LoIndx.Array.Elem(ia,4) to PD.HiIndx.Array.Elem(ia,4), that is, over the columns in the cross talk data for array ia. The index "ia" is described above. Index, n=1,2
Units/representation	SC.Array.CrossTalkQ2.*Ia*(j,k,n) is single float SC.Array.CrossTalkQ2.*Ia*(j,k,1 (dimensionless) SC.Array.CrossTalkQ2.*Ia*(j,k,2) (seconds)
Treatment/usage	See SC.Array.CrossTalkQ1.*Ia*(j,k,n) for the basic discussion of the cross talk data. SC.Array.CrossTalkQ2.*Ia*(j,k,n) is the second quadrant counter part to SC.Array.CrossTalkQ1.*Ia*(j,k,n). If its data values are basically a reflection of the data in SC.Array.CrossTalkQ1.*Ia*(j,k,n) about the first column, SC.Array.CrossTalkQ2.*Ia*(j,k,n) need not be populated and the values will be implied. If that symmetry does not exist, SC.Array.CrossTalkQ2.*Ia*(j,k,n) should be populated to supply the correct cross talk data for the second quadrant.

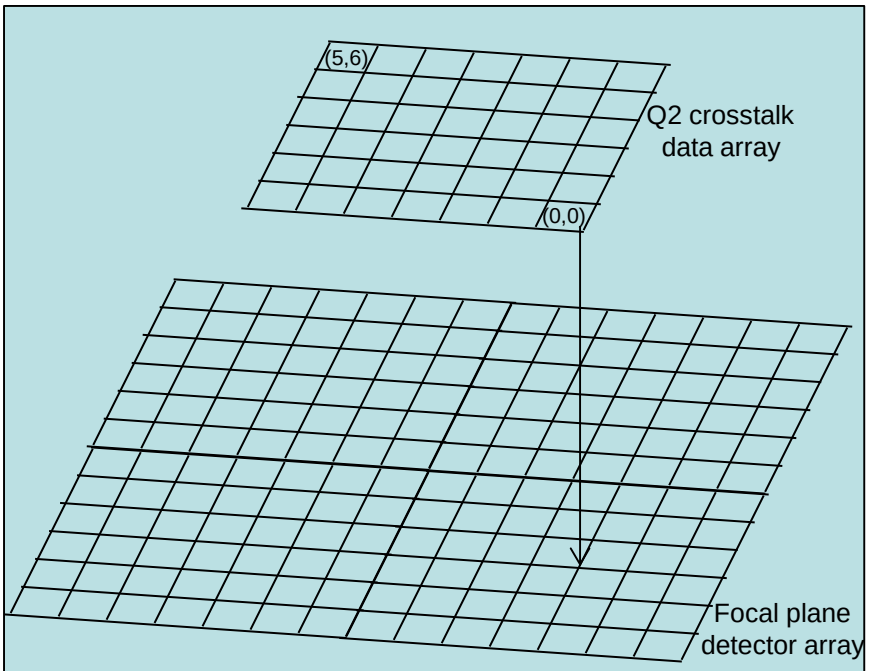


Illustration of a 6 row, 7 column crosstalk data grid with its origin "pixel" applied to an arbitrary detector on the focal plane.

SC.Array.CrossTalkQ2.*Ia*(j,k,2) is the delay in seconds of the response in the detector element referenced by (j,k) when no photon (flux) is incident after the response in the element referenced by (0,0) when a unit of photon flux is incident on that element. We are treating the *pattern* of time delay as homogeneous across the focal plane by supplying only one SC.Array.CrossTalkQ2.*Ia*(j,k,2) for an array. It is assumed that the delays in cross talk response are independent of slow time. Since no element can both have photon flux incident on it and not have photon flux incident on it, we require that SC.Array.CrossTalkQ2.*Ia*(0,0,2)= 0.000000E+00.

Governing quality metrics

See Also PD.HiIndx.Array.Elem(ia,3) SC.Array.CrossTalkQ1.*Ia*(j,k,n)
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.5.4 SC.Array.CrossTalkQ3.*Ia*(j,k,n) [L1,2]

Brief Definition SC.Array.CrossTalkQ3.*Ia*(j,k,n) provides cross talk levels and delays for the lidar's detector array.

indices Substring, "*Ia*", acts as an index and the character(s) represents integers in the range from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally

from 1 to the number of focal plane arrays. This is an evaluated substring so that the length of the cross talk data arrays can be adjusted to the needs of each array.
Index, j , ranges from PD.LoIndx.Array.Elem(ia,3) to PD.HiIndx.Array.Elem(ia,3), that is, over the rows in the cross talk data for array ia. The index "ia" here is the integer determined by the value of the substring, "*Ia*". For example, if the substring,"*Ia*", is populated with the characters, "02", then the corresponding value of ia is 2.
Index, k , ranges from PD.LoIndx.Array.Elem(ia,4) to PD.HiIndx.Array.Elem(ia,4), that is, over the columns in the cross talk data for array ia. The index "ia" is described above.
Index, n=1,2

Units/representation

SC.Array.CrossTalkQ3.*Ia*(j,k,n) is single float
SC.Array.CrossTalkQ3.*Ia*(j,k,1 (dimensionless)
SC.Array.CrossTalkQ3.*Ia*(j,k,2) (seconds)

Treatment/usage

See SC.Array.CrossTalkQ1.*Ia*(j,k,n) for the basic discussion of the cross talk data.
SC.Array.CrossTalkQ3.*Ia*(j,k,n) is the third quadrant counter part to SC.Array.CrossTalkQ1.*Ia*(j,k,n). If its data values are basically a reflection of the data in SC.Array.CrossTalkQ1.*Ia*(j,k,n) through the origin, SC.Array.CrossTalkQ3.*Ia*(j,k,n) need not be populated and the values will be implied. If that symmetry does not exist, SC.Array.CrossTalkQ3.*Ia*(j,k,n) should be populated to supply the correct cross talk data for the second quadrant.

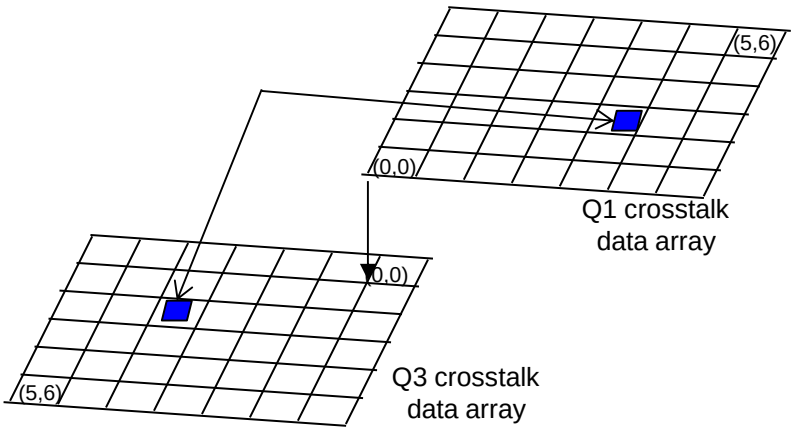


Illustration of 3th quadrant cross talk data implied when only 1st quadrant data is populated. The "origin pixels" overlap and one pair of corresponding grid positions determined by reflection across the row axis are indicated.

SC.Array.CrossTalkQ3.*Ia*(j,k,2) is the delay in seconds of the response in the detector element referenced by (j,k) when no photon (flux) is incident after the response in the element referenced by (0,0) when a unit of photon flux is incident on that element. We are treating the *pattern* of time delay as homogeneous across the focal plane by supplying only one SC.Array.CrossTalkQ3.*Ia*(j,k,2) for an array. It is assumed that the delays in cross talk response are independent of slow time. Since no element can both have photon flux incident on it and not have photon flux incident on it, we require that SC.Array.CrossTalkQ3.*Ia*(0,0,2)= 0.000000E+00.

Governing quality metrics

See Also PD.HiIndx.Array.Elem(ia,3) SC.Array.CrossTalkQ1.*Ia*(j,k,n)
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.5.5 SC.Array.CrossTalkQ4.*Ia*(j,k,n) [L1,2]

Brief Definition SC.Array.CrossTalkQ4.*Ia*(j,k,n) provides cross talk levels and delays for the lidar's detector array.

indices Substring, "*Ia*", acts as an index and the character(s) represents integers in the range from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays. This is an evaluated substring so that the length of the cross talk data arrays can be adjusted to the needs of each array.
Index, j , ranges from PD.LoIndx.Array.Elem(ia,3) to PD.HiIndx.Array.Elem(ia,3), that is, over the rows in the cross talk data for array ia. The index "ia" here is the integer determined by the value of the substring, "*Ia*". For example, if the substring,"*Ia*", is populated with the characters, "02", then the corresponding value of ia is 2.
Index, k , ranges from PD.LoIndx.Array.Elem(ia,4) to PD.HiIndx.Array.Elem(ia,4), that is, over the columns in the cross talk data for array ia. The index "ia" is described above.
Index, n=1,2

Units/representation SC.Array.CrossTalkQ4.*Ia*(j,k,n) is single float
SC.Array.CrossTalkQ4.*Ia*(j,k,1 (dimensionless)
SC.Array.CrossTalkQ4.*Ia*(j,k,2) (seconds)

Treatment/usage See SC.Array.CrossTalkQ1.*Ia*(j,k,n) for the basic discussion of the cross talk data.
SC.Array.CrossTalkQ4.*Ia*(j,k,n) is the fourth quadrant counter part to SC.Array.CrossTalkQ1.*Ia*(j,k,n). If its data values are basically a reflection of the data in SC.Array.CrossTalkQ1.*Ia*(j,k,n) around the first row axis, SC.Array.CrossTalkQ4.*Ia*(j,k,n) need not be populated and the values will be implied. If that symmetry does not exist, SC.Array.CrossTalkQ4.*Ia*(j,k,n)

should be populated to supply the correct cross talk data for the second quadrant.

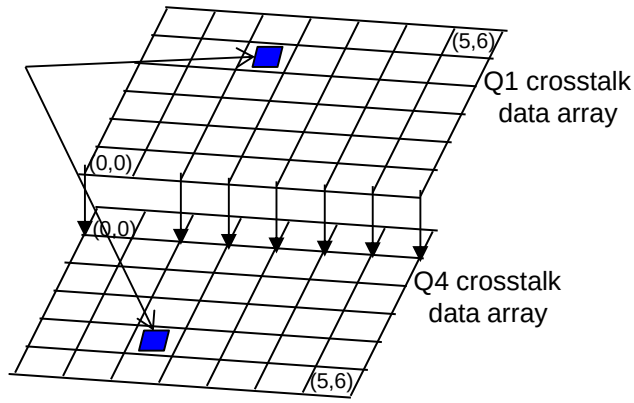


Illustration of 4th quadrant cross talk data implied when only 1st quadrant data is populated. The axis rows overlap and one pair of corresponding grid positions determined by reflection across the row axis are indicated.

SC.Array.CrossTalkQ4.*Ia*(j,k,2) is the delay in seconds of the response in the detector element referenced by (j,k) when no photon (flux) is incident after the response in the element referenced by (0,0) when a unit of photon flux is incident on that element. We are treating the *pattern* of time delay as homogeneous across the focal plane by supplying only one SC.Array.CrossTalkQ4.*Ia*(j,k,2) for an array. It is assumed that the delays in cross talk response are independent of slow time. Since no element can both have photon flux incident on it and not have photon flux incident on it, we require that SC.Array.CrossTalkQ4.*Ia*(0,0,2)= 0.000000E+00.

Governing quality metrics

See Also PD.HiIndx.Array.Elem(ia,3) SC.Array.CrossTalkQ1.*Ia*(j,k,n)
SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.5.6 SC.Array.FilterDat(ifc,j) [L1,2,3,4,5]

Brief Definition SC.Array.FilterDat(ifc,j) specifies the spectral response for the individual optical band pass filters.

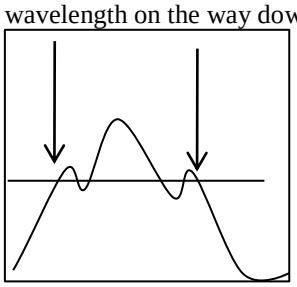
indices Index, “ifc”, ranges from PD.LoIndx.Array(2) to PD.HiIndx.Array(2), that is, nominally from 1 to the number of spectral filters described.
Index, j, ranges from PD.LoIndx.Array(3) to PD.HiIndx.Array(3), that is, nominally from 1 to the number of specified filter parameters. Currently

defined up to PD.HiIndx.Array(3)=9.

Units/representation: Float, units vary with value of j, see below

Treatment/usage: j=1 → wavelength of the peak transmittance of the filter (λ_{peak}), meters
j=2 → longest long wave side pass band half power wavelength (λ_{3dB+}), meters

If there is a sidelobe like shoulder where the filter response drops below -3db, but the dip is well above the next minimum, we would measure the -3dB point at the longer wavelength on the way down to a more significant frequency cutoff.



j=3 → shortest Short wave side pass band half power wavelength (λ_{3dB-}), meters
• j=4 → transmittance at the filter peak (T_{peak}), dimensionless fraction from 0.0E+0 to 1.0E+0
• j=5 → long wavelength side attenuation rate ($Atten_{rate+}$) in dB/octave smoothed attenuation between λ_{3dB+} and rejection band, dimensionless
• j=6→ short wavelength side attenuation rate ($Atten_{rate-}$) in dB/octave smoothed attenuation between λ_{3dB-} and rejection band, dimensionless
• j=7 → max transmittance in rejection bands (T_{rej}), dimensionless fraction from 0.0E+0 to 1.0E+0
• j=8→ $10\log_{10}$(fraction of solar energy in pass band within the -60dB focal naked detector response spectrum), (Solar_leak) dimensionless
• j=9→ 10 $\log_{10}$ (ratio of solar energy in pass band to energy in reject bands integrated out to detector -60dB response wavelengths) (SNR) dimensionless

Governing quality metrics: Percentage error from truth

See Also: PD.HiIndx.Array

20.5.7 SC.Array.OptclCent(ia,j) [L1,2]

Brief Definition: SC.Array.OptclCent(ia,j) is the position of the pierce point of the optical axis on the sensing plane.

indices

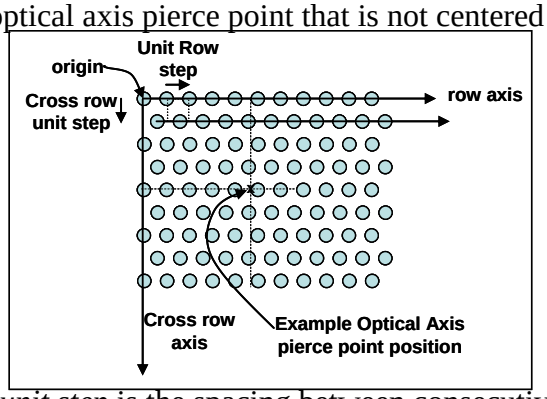
Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays[201]
j=1--> row value of Optical Axis pierce point at detector focal plane
j=2--> "cross row" value of Optical Axis pierce point of detector Focal plane

Units/representation

Float, Dimensionless

Treatment/usage

This is a position in a Cartesian system laid over the sensor focal plane device. It is given by two values. The first is the number of (row) unit steps along a row direction. A unit step in the row direction is the center to center detection element spacing in a direction that has uniformly spaced detectors. The second is the number of "cross row" unit steps in a direction orthogonal to the row direction. The term "cross row" has been introduced so that positions with respect to a general lattice of detectors may be described by Cartesian coordinates, even if the detector array does not possess both rows and columns. [It may even be composed of chunks of lattices, as in the case of a compound array.] Lattice spacings are used rather than absolute metric distances. The scheme reduces to row –column coordinates when the lattice is a rectangular grid. The ability to define positions in fractional steps permits specifying an optical axis pierce point that is not centered on an element.



The *cross row unit step* is the spacing between consecutive rows. The *origin* of the focal plane Cartesian frame is defined as the center of the detector that is upper most to the left when viewing the focal plane along incident rays with the row direction rotated so that it orients in the left to right direction. The *row axis* is the line through the origin in the row direction. The cross row axis is the line through the origin in the cross row direction. For a detector array that is symmetric under a 90° rotation, an arbitrary choice of what the row direction is may be made when populating this field. However, certain other data must be populated consistently with this choice. This is the point

[201] Note that a multichannel lidar will have a specific array assigned to each channel. It is permitted that the lidar have only one array and more than one channel.

whose position is given by SC.Pulse.SensorPos.*ChNum*.*TileNum*(j,ip).

Governing quality metrics: Accuracy of data=pierce point error/unit row step, error/unit cross row step

See Also: SC.Pulse.SensorPos.*ChNum*.*TileNum*(j,ip) SC.Array.Chunks(ich,j) SC.Array.LatticeDesc(j)

20.5.8 SC.Array.PolDat(ipc,j) [L1,2,3,4,5]

Brief Definition: SC.Array.PolDat(ipc,j) specifies the response for each individual polarizing receive chain filter used in collecting the data for the current dataset.

indices: Index, ipc, ranges from PD.LoIndx.Array(4) to PD.HiIndx.Array(4), that is, nominally from 1 to the number of polarizers described. Values of ipc will be cited by SC.Channlzn.Descr(j,4) to define channels.

Index, j, ranges from PD.LoIndx.Array(5) to PD.HiIndx.Array(5), that is, nominally from 1 to the number of specified polarizer parameters. Currently defined up to PD.HiIndx.Array(5)=59 [if PD.LoIndx.Array(5) is set to 1].

Units/representation: Various, float

Treatment/usage: When a lidar uses polarization filters to create the recorded data for a channel, we need to describe the receiver polarization for each channel. The net optical bench polarization effect for a single receiver optical path is given by the ipcth list (over index, j) of SC.Array.PolDat(ipc,j).

We support polarizers capable of analyzing incident light into arbitrary general elliptical polarizations. This covers the special cases of linear polarizations and circular polarizations, which are likely implementation designs. This means that we will describe the filters by their Mueller matrices, which convert arbitrary polarizations to other arbitrary polarizations when those polarizations are represented as Stokes vectors according to

$$\begin{pmatrix} I' \\ Q' \\ U' \\ V' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \begin{pmatrix} I \\ Q \\ U \\ V \end{pmatrix}$$

where the input stokes vector is the unprimed column vector and the output column vector is the primed column vector.
The Stokes vector is defined in terms of the light wave parameters by

$$S_0 = \mathrm{I} = \mathrm{E}_{0x}^2 + \mathrm{E}_{0y}^2$$
$$S_1 = \mathrm{Q} = \mathrm{E}_{0x}^2 - \mathrm{E}_{0y}^2$$
$$S_2 = \mathrm{U} = 2\mathrm{E}_{0x}\mathrm{E}_{0y}\cos\varepsilon$$
$$S_3 = \mathrm{V} = 2\mathrm{E}_{0x}\mathrm{E}_{0y}\sin\varepsilon$$

The phase of the y-component of the electric field exceeds that of the x-component of the electric field by ε.

In terms of the polarization ellipse, the angle between the major axis of the ellipse and the x-axis is $\theta = \frac{1}{2}\tan^{-1}\frac{S_2}{S_1}$. For the case of linear polarization, the major axis lies along the E vector. When the electric vector lies in the x-direction, θ =0 degrees.

The x-direction for polarization referencing is defined to be the array major axis direction whose ECEF orientation is provided for each pulse in SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) for ia= PD.LoIndx.Array(1). To be clear, all channels, regardless of which array is used, use the first array's major axis as the basis for receiver polarization x-direction definition. The polarization filters are thus described with respect to a sensor based vector. This direction may change in the earth frame on a pulse by pulse basis, but all comparisons of polarizing filters for a single pulse are based on a single reference direction for that pulse. Furthermore, all polarization filter data for each pulse is given in the sensor frame, and thus pulse to pulse polarization comparisons may involve rotations of the pulse independent polarization parameterization to be found in specific polarization filter Mueller matrices. The Mueller matrices, M, can be rotated by an angle, ϕ, about the axis of the optical component via the transformation

$$M' = R(-\phi)MR(\phi)$$

Where

$$R(\phi) = 0\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\phi & \sin 2\phi & 0 \\ 0 & -\sin 2\phi & \cos 2\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Polarizer parameters for polarizer ipc:

- j =1 gives the wavelength, λ, of peak polarization selectivity (λ_0) [meters]
- j =2 A positive number indicates that the polarization filter is separate

physically from a possible spectral filter. A negative number indicates that the polarization data applies to the spectral filter and spectral and polarization filtering is combined. [dimensionless]

• j =3 -- Gives the orientation of the polarization ellipse major axis for the peak filter transmission Stokes vector. This is measured [in degrees, counter clockwise looking into the sensor along the optical axis] with respect to the direction of the major (row) axis of the array indicated by "ia"=PD.LoIndx.Array(1). Note that this is principally a convenience item since the orientation of the polarization ellipse passed by the ipcth filter can be derived from the Mueller matrix elements in SC.Array.Poldat(ipc, 20 to 35). We will require that the value populated here be within 0.1 degree of the value derived from the Mueller matrix elements.

• j= 4 through 19 are, respectively, elements m_{11}, m_{12}, m_{13}, m_{14}, m_{21}, m_{22}, m_{23}, m_{24}, m_{31}, m_{32}, m_{33}, m_{34}, m_{41}, m_{42}, m_{43}, m_{44} of <u>the ideal Mueller matrix</u> for polarization filter, ipc. The "ideal" Mueller matrix is the intended target filter property. For example, to achieve a linear polarization at -45 degrees, m_{11} and m_{33} are set to 0.5 and m_{13} and m_{31} are set to -0.5 while all other elements are set to zero. The Mueller matrix geometric reference is chosen to establish the x-component of the light wave electric vector as in the major axis direction of the first array (ia= PD.LoIndx.Array(1)) and the y-component of the electric vector is orthogonal to the x- direction in the focal plane and forming a right handed system with the outward looking optical axis.[202]

• j=20 through 35 are, respectively, the measured values of the elements m_{11}, m_{12}, m_{13}, m_{14}, m_{21}, m_{22}, m_{23}, m_{24}, m_{31}, m_{32}, m_{33}, m_{34}, m_{41}, m_{42}, m_{43}, m_{44} of <u>the actual laboratory calibrated Mueller matrix</u> at λ_0 for polarization filter, ipc. Typically there will be small values where there were zeros in the ideal Mueller matrix and the .5 valued elements in the ideal matrix will be slightly different in the actual Mueller matrix as well. The expected degree of cross talk between polarization channels at λ_0 can be derived using this information.

• j=36 through 51 are, respectively, the measurement uncertainties (standard deviation values) for the elements m_{11}, m_{12}, m_{13}, m_{14}, m_{21}, m_{22}, m_{23}, m_{24}, m_{31}, m_{32}, m_{33}, m_{34}, m_{41}, m_{42}, m_{43}, m_{44} of <u>the actual laboratory calibrated Mueller matrix</u> at λ_0 for polarization filter, ipc. High confidence percentile conversion factors corresponding to these standard deviations may be populated into ES.Swath.HiConfdncP_Fac(i,j,k).

[202] For a sensor design that might use multiple groups of polarimetric channels each working in somewhat different look directions, the meaning of the "first array", must be extended so that in addition to the array indicated by PD.LoIndx.Array(1), the lowest "ia" value for each group is used to establish the reference for the x and y electric vector components used in defining the Stokes/Mueller framework for that group.

For the following, we define a fitting function to describe the degree to which wavelength mismatch can be ambiguous with polarization mismatch in determining the intensity of transmission through the polarization filters. This information would be used to characterize polarization based segmentation errors and perhaps need not be exceptionally precise. We supply a single factor, $\rho(\tilde{\lambda})$, for a value of $\tilde{\lambda}$ =|λ-λ_0| (per λ on the higher and lower wavelength side of λ_0) to act as a multiplicative correction to the *attenuation level* of orthogonal polarizations due to action of the actual Mueller matrix at λ_0. This serves to indicate the spectrally driven reduction in filter selectivity. That is, $\rho(\tilde{\lambda})$ characterizes the wavelength dependence of cross talk. We provide a set of coefficients for the longer and shorter wavelength directions, since symmetry is not very likely (especially if dichroics are in use)

$$\rho(\tilde{\lambda}) = \left(a + \frac{b-a}{\left(1+Qe^{c\tilde{\lambda}}\right)^{1/d}} \right)$$

Where:

a controls the large lambda twiddle behavior

b controls the initial curvature of the spectral dependence

c controls peak rate change due to the spectral dependence

d controls approach to large wavelength deviation from λ_0

Define the wavelength dependent attenuation of orthogonal polarizations by a filter to be given by

$$att(\lambda) = \frac{I(\lambda_0) - \hat{I}(\lambda)}{I(\lambda_0)}$$

so that zero attenuation results when the orthogonal polarization at λ has the same intensity as the selected polarization at λ_0, i.e. , $I(\lambda_0) = \hat{I}(\lambda)$. Perfect attenuation is when the intensity of the passed orthogonal polarization is zero, i.e. att(λ)=1. We produce a $\rho(\lambda)$ so that

$$att(\lambda) = att(\lambda_0)\rho(\lambda)$$

and $\rho(\lambda_0)$=1. A value of 0 for $\rho(\lambda)$ leads to *att*(λ)=0 , that is $I(\lambda_0) = \hat{I}(\lambda)$, which is very poor performance.

• j=52 through 55 are, respectively, the coefficients a,b,c,d of a fitting function to the wavelength dependence of the ipcth polarization filter for wavelengths shorter than λ_0 . (high frequency side) [wavelength in meters]
• j=56 through 59 are, respectively, the coefficients a,b,c,d of a fitting function to the wavelength dependence of the ipcth polarization filter for wavelengths longer than λ_0 . (low frequency side) [wavelength in meters]

If SC.Array.Poldat(ipc,j) is not populated, it is not possible to populate SC.Channlzn.Descr(j,4) with a value other than 0. Any data set with SC.Channlzn.Descr(j,k) unpopulated or SC.Channlzn.Descr(j,4) set to zero is to be presumed to have data of unknown polarization.

Governing quality metrics

See Also SC.Channlzn.Descr(j,4) PD.HiIndx.Array ES.Swath.HiConfdncP_Fac(i,j,k)

20.5.9 SC.Array.RayDirs.*Ia*(icase,ie,j) [L1,2]

Brief Definition SC.Array.RayDirs.*Ia*(icase,ie,j) gives the look direction relative to the optical axis as a direction cosine pair.

indices *Ia*, though part of a tag name, acts as an index. It indicates which uniquely named parameter carries the data as an array of values in a named block of data. *Ia* acts as the row in which a data item is found in a 4-D matrix. Name substring, *Ia*, is an alphanumeric (character) coding of an integer that ranges from SC.Array.LoIndx(1) to SC.Array.HiIndx(1), that is, generally from 1 to the number of arrays. This device permits the range of ie to vary from array to array.

Index, icase, ranges from PD.LoIndx.Array(10) to PD.HiIndx.Array(10), that is, generally from 1 to the number of different optical axis steering direction case values. This permits there to be several element look direction maps, as might be appropriate for the case when the actual element look directions depend on the direction to which the optical axis has been steered.

Index, ie, ranges from PD.LoIndx.Array.Elem(ia) to PD.HiIndx.Array.Elem(ia), that is, generally from 1 to the number of array elements for array ia. The number of elements is provided directly by SC.Array.LatticeDesc(ia,1). Certain implementations may choose to base array indexing at 0. The physical association of hardware detector elements to the element index sequence “ie” must be consistent throughout any single

data set.

Index, j, ranges from 1 to 5

Units/representation Double Float, dimensionless

Treatment/usage For each array (indexed by ia) and for each optical axis direction case (indexed by icase), each element has a separate look direction associated with the ray on its field of view cone axis. This direction is given by two direction cosines.

The look direction of individual elements may depend on the degree to which the array axis has been steered. This could be the result of compound steering mechanisms, refraction by a domed window for the lidar head, etc. If this is the case, we present the element look directions in separate layers, one devoted to each array axis steering direction case. Rather than attempt to parameterize a function for each element that serves all optical benches over the full range of axis steering, we populate these array element look direction layers with discreet and distinct lists of look directions. The number of these layers is budgeted to the permitted geolocation error for the lidar system. The look direction value for an element in a given layer applicable to a given pulse is used without interpolation. The index, icase, indicates the layer element look direction data associated with a specific axis steering range. A pulse indexed parameter, SC.Array.DirCaseNum(ip) associates the correct choice of icase layer with each pulse.

The direction cosines quantify the angle between the element field of view cone axis ray and the orthogonal array directions.

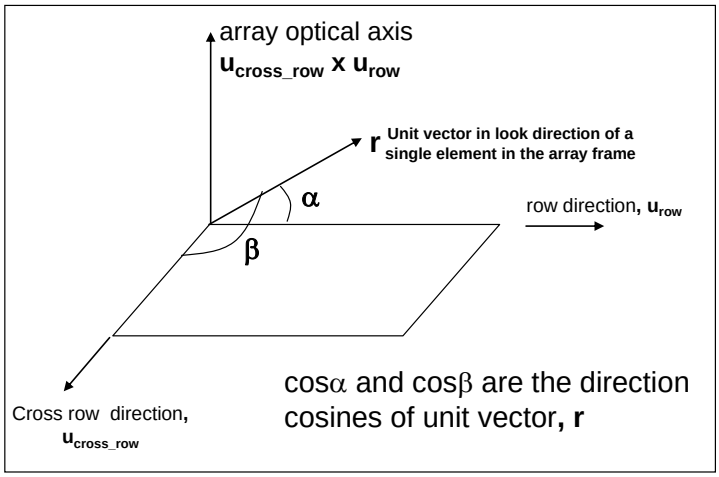
For j=1, the direction cosine is provided for the angle, α, measured from the row direction as indicated in the figure below.

For j=2, the direction cosine is provided for the angle, β, measured from the cross row direction as indicated in the figure below.

SC.Array.RayDirs.*Ia*(icase,ie,3) gives the effective major axis field of view of the element, “ie”, in case, “icase”. The field of view of a detector element is modeled as an equivalent elliptical cone of photon sensitivity. The element is at the vertex of the cone and 95% of received photo power reported by the detector is received from the space subtended by this cone.

SC.Array.RayDirs.*Ia*(icase,ie,4) gives the effective minor axis field of view of the element, “ie”, in case, “icase”. The field of view of a detector element is modeled as an equivalent elliptical cone of photon sensitivity. If the major and minor axes of the element FOV cone are the same, the cone is circular and the orientation in SC.Array.RayDirs.*Ia*(icase,ie,5) can be ignored.

SC.Array.RayDirs.*Ia*(icase,ie,5) gives the orientation of the major axis of the element field of view cone as an angle measured counter clockwise from the projection into the receiver optical axis normal plane of the sensor frame x axis (or y-axis if the x axis, is parallel to the look direction of the receiver).



Note on the association of a value of index ie, to specific detector elements:
As stated in the above index cell, ie is an index that runs over the individual receptors in the focal plane array. The application that first writes the SC.Array.RayDirs(icase,ie,j) data array into a data set is rather free to choose how to associate a single number to each element. Because "ie" is an index, the set of numbers that are used to label the individual elements must be consecutive. A very reasonable (but not mandated) method to linearize a 2-D, possibly chunked [see SC.Array.Chunks(ich,j)] , focal plane array is to assign ie =1,2,3,… to elements along consecutive rows and in chunk order.

Governing quality metrics: Uncertainties provided

See Also: SC.Array.RayDirsSigmas(ia,icase,j) SC.Array.LatticeDesc(ia,j). SC.Array.DirCaseNum(ip) PD.HiIndx.Array SC.Pulse.SensorAxisX.*TileNum*(ip,j)

20.5.10 SC.Array.RayDirsSigmas(ia,icase,j) [L1,2]

Brief Definition: SC.Array.RayDirsSigmas(ia, icase,j) supplies the uncertainty in the look direction for all of the elements in a steering direction case for a given array.

indices: Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is,

nominally from 1 to the number of focal plane arrays.
Index, icase, ranges from PD.LoIndx.Array(6) to PD.HiIndx.Array(6), that is generally from 1 to the number of different optical axis steering direction case values. This permits there to be separate uncertainties for each element look direction map layer.
Index j =1,2 for the direction cosine uncertainties.

Units/representation: Float, dimensionless

Treatment/usage: The direction of a ray in the array frame is given by a pair of direction cosines. The uncertainty in the look direction is given by a "delta direction cosine" for each angle in the pair. The direction cosine for each axis is thus given by (value ± uncertainty) where "value" is obtained from SC.Array.RayDirs(ia, icase,ie,j) and the "uncertainty" is given by SC.Array.RayDirsSigmas(ia, icase,j).
j=1 has the uncertainty in the direction cosine referenced to the row axis.
j=2 has the uncertainty in the direction cosine referenced to the cross_row axis.

Governing quality metrics: Values are true/not true

See Also:

20.5.11 SC.Array.LatticeDesc(ia,j) [L1,2]

Brief Definition: SC.Array.LatticeDesc(ia,j) gives the total number of detecting elements over the lattice chunks and high level lattice chunk description

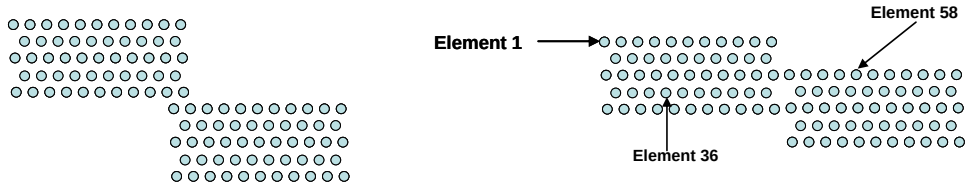
indices: Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays.
Index, j, ranges from PD.LoIndx.Array(7) to PD.HiIndx.Array(7) that is over the number of lattice description parameters, currently defined to be 3.

Units/representation: Dimensionless, UInt32

Treatment/usage: The focal plane sensing array could be a composite of multiple chips, each a separately manufactured lattice of detectors. Each of the chips, here called lattice chunks, would be mounted physically adjacent to each other. All focal plane lattices are such objects since it is to be understood that that the number of chunks could be just one. For each array, indexed by ia, we have the descriptors:
j=1-- total number of elements across all chunks
j=2-- number of array chunks
j=3-- code for chunk alignment
1=> full row overlap } makes longer or wider

2=> full cross row overlap parallelogram/rectangle coverage
3=> partial row direction overlap
4=> partial overlap in cross row direction
5=> partial overlap in both row and cross row direction
6=> no overlap in row or cross row direction

Examples of compound arrays of multiple chunks of detector lattices are shown below.



Each individual detecting element in such a lattice has a unique index value which is the count along rows by lattice chunk.
A single compound array has a single value of array index, ia, even if it has several chunks. To warrant a separate array index, an array, be it composed of one or more lattice chunks, must participate in a separate channelization and thus have a separate polarization, spectral filter, illuminator, or look direction from the other arrays.

Governing quality metrics: Values are must be correct.

See Also: SC.Array.Chunks(ich,j) SC.Array.OptclCent(j) PD.HiIndx.Array

20.5.12 SC.Array.Chunks(ia,ichk,j) [L1,2]

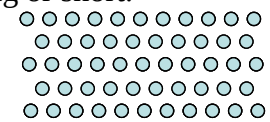
Brief Definition: SC.Array.Chunks(ia,ichk,j) describes the parts of a compound array lattice.

indices: Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays.
Index, ichk, ranges from 1 to SC.Array.LatticeDesc(ia,2), that is over the individual array chunks. The physical association with hardware chunks to the chunks index sequence ichk must be consistent throughout any single data set.
Index, j, ranges from PD.LoIndx.Array(10) to PD.HiIndx.Array(10) that is over the number of lattice chunk description parameters, currently defined to be 6.

Units/representation: Dimensionless, UInt16

Treatment/usage: The notion of a chunk of a composite detector array is defined in the paragraph defining SC.Array.LatticeDesc(ia,j). Each chunk of the possibly composite detector array is a general lattice of detecting elements. The

general lattice is composed of rows of uniformly spaced elements in a line. The rows need not all have the same number of elements. In fact, for triangular lattices, no only do the columns appear to be staggered, but alternating rows can be long or short.



The simplest lattice is a rectangular grid. In that case, the number of elements in short rows and long rows is the same and elements align in linear columns as well as rows.

There are currently five values defining a chunk

j=1 number elements in a short row

j=2 number of elements in a long[203] row

j=3 number of short rows in array chunk

j=4 number of long rows in array chunk

j=5 =0 for rectangular lattice, =1 for triangular lattice

j=6 =0 for first row is a short row, =1 for first row is a long row (viewing the detector array as a 4th quadrant array with the origin in the upper left); short and long rows interlaced alternately with the number of consecutive rows of same length =1. SC.Array.Chunks(ia,ichk,6) will be ignored if SC.Array.Chunks(ia,ichk,5) = 0.

Governing quality metrics: Values must be correct

See Also: SC.Array.LatticeDesc(j) PD.HiIndx.Array(j)

20.5.13 SC.Array.CurrentResp.Dat(ia,j) [L1,2]

Brief Definition: SC.Array.CurrentResp.Dat(ia,j) carries waveform clarity metrics.

indices: Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays.

Index, j, ranges from PD.LoIndx.Array(12) to PD.HiIndx.Array(12), that is over the parameters for specifying the fidelity of the waveform response to incident photon flux.

Units/representation: Various, Float

Treatment/usage: We characterize the clarity of the waveform response by the lidar sensor via a temporal response and a noise level.

The temporal response is described by a series of parameters indicating the measured fast time response of the data that can be recorded from the lidar receiver. Since an ideal impulse is not available to measure the system

[203] If all rows have equal lengths, the j=1,2 values will be the same.

response, and approaches to impulse response measurement have various levels of fidelity that could require a large amount of metadata to characterize, we provide data on the test stimulus chosen to probe the performance limits of the sensor as well as the lidar receiver response being claimed. We represent a test pulse used to measure/characterize the sensor response as parameterized by a leading edge, rising side Gaussian and a trailing edge, tail Gaussian that meet at the pulse peak. A similar parameterization of the sensor response is provided.
We give a high level characterization in terms of the half power widths of the test impulse peak and the response peak as well as the leading and trailing Gaussian widths for the test impulse and the response. Some measurement methods/waveforms may not be able to provide more information than the half power pulse widths (j=1,2), so these are separate data items that can be provided when the rising and falling time durations can't be separately resolved. When the rise and fall time durations are measurable, they should sum to the values in the j=1,2 data slots of SC.Array.CurrentResp.Dat(ia,j). The CMMD supports the capability to provide only the net width without leading and trailing components to avoid misleading users by making the two parts of the wave form half power width equal when the balance is, in fact, unknown.

It is to be understood that the values given here are the result of averaging a large number of trials so that the values supplied can be used to evaluate the reliable fidelity of waveform data vectors [LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)] in the data set.

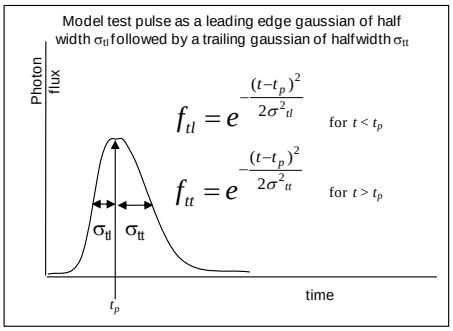
SC.Array.CurrentResp.Dat(ia,1)= the half power time "width", in seconds, of the recorded photocurrent response to a pulse of incident photons of duration width SC.Array.CurrentResp.Dat(ia,2) that drives the photocurrent to greater than 20dB above the noise floor but not into saturation.

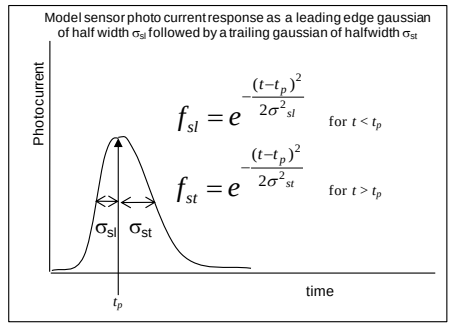
SC.Array.CurrentResp.Dat(ia,2)= the duration , in seconds, measured at the half power level of the mean test pulse used to characterize the sensor's captured photocurrent wave form response to incident photon flux.

SC.Array.CurrentResp.Dat(ia,3)= test pulse leading edge Gaussian fit width (in seconds) parameter, $\sigma_{tl.}$ This characterizes the part of the test pulse preceding the peak and is fit to the rising edge from the peak to the -10dB point.
SC.Array.CurrentResp.Dat(ia,4)= test pulse tail Gaussian fit width (in

seconds) parameter, $\sigma_{tt.}$ This characterizes the part of the test pulse following the peak and is fit to the tail from the peak to the -10dB point.

SC.Array.CurrentResp.Dat(ia,5)= leading edge Gaussian fit width (in seconds) parameter, σ_{sl} of the sensor response to a test pulse. This characterizes the part of the impulse response preceding the peak and is fit to the rising edge from the peak to the -10dB point.
SC.Array.CurrentResp.Dat(ia,6)= trailing edge Gaussian fit width (in seconds) parameter, σ_{st} of the sensor response to a test pulse. This characterizes the part of the impulse response following the peak and is fit to the tail from the peak to the -10dB point.





SC.Array.CurrentResp.Dat(ia,7)= ratio of the sensor noise level to the reported current quantization step at the bottom of the photocurrent dynamic range in a waveform data vector. If this value is greater than one, then the noise should be clearly visible in the recorded data. If this value is less than one, the noise is below the digitization step at the low magnitude end of the data range.

For Geiger mode collections, this item should be unpopulated.
If only some of the measurements conveyed by SC.Array.CurrentResp.Dat(ia,j) are populated, the missing values will be indicated by using a negative number.

Governing quality metrics
Must fairly represent the lidar sensor performance. Uncertainty to value ratio.

See Also
ES.Point.IntensyFlg PD.HiIndx.Array(12)
LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)]

20.5.14 SC.Array.CurrentResp.Flg(ia) [L1,2]

Brief Definition
SC.Array.CurrentResp.Flg(ia) indicates whether the recorded data from the sensor reflects that a sensor photocurrent amplifier has a linear or log (that is, dB) response.

indices	Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays.
Units/representation	Dimensionless, UInt8
Treatment/usage	SC.Array.CurrentResp.Flg(ia) tells how to interpret values of LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) when vector quantization is not employed. When vector quantization is employed, the decode table will implicity eliminate the need for this parameter and SC.Array.CurrentRespFlg(ia) should not be populated. SC.Array.CurrentResp.Flg(ia) =1 or unpopulated: Values of LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) are proportional to photo current (linear response to inferred incident power flux) SC.Array.CurrentResp.Flg(ia) =2: Values of LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) are proportional to the logarithm of detected photo current. For Geiger mode collections, this item should be unpopulated. Not only does working with the logarithm of the incident intensity reduce the dynamic range, but when the uncertainty in intensity measurement is proportional to the intensity, the log makes it possible for the reported increment to be have a constant ratio with the uncertainty.
Governing quality metrics	Must fairly represent the lidar collection and processing
See Also	ES.Point.IntensyFlg

20.5.15 SC.Array.DirCaseNum.*TileNum*(ip,j) [L1,2]

Brief Definition	SC.Array.DirCaseNum.*TileNum*(ip,j) provides the value of the steering direction case that applies to each pulse.
indices	Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles. The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.DirCase.*TileNum*.Flg set to zero or not populated. When IntTileMap.DirCase.*TileNum*.Flg >0, i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. The SC.Array.DirCaseNum *TileNum*(ip,1) in this case carries the number of times the value in SC.Array.DirCaseNum.*TileNum*(ip,2) is repeated consecutively. This index supports repetition factor use to reduce data volume.

	Consider the effective data vector created by logical concatention of the data vectors of SC.Array.DirCaseNum.*TileNum*(ip,j) for all consecutive values of *TileNum* that are in the tags, SC.Array.DirCaseNum.*TileNum*. This would be a list of direction case values for each pulse of the collection operation represented in the current data set. The list has a direction case value for each pulse, in ascending order, from the first IPP (IPP number 0) to the last for the data in the data set. The point here is that "ip" is an index, not a stored data item in this definition paragraph. The range of the index, usually defined via a Lo/HiIndx, is here governed by PD.TileMap.PulseRange(j_tile,1) and PD.TileMap.PulseRange(j_tile,2).
Units/representation	Dimensionless, UInt8
Treatment/usage	Due to possible optical path effects coupling with possibly large scanning angles for the lidar head, the look direction for any single detector element on the focal plane may vary over a number of pulses. If a lidar is operated in a mode where it is necessary to compute XYZ from ray direction and range using pulse dependent ray directions, SC.Array.DirCaseNum(ip,j) is used to supply the correct choice of element look direction map. This data item would be most likely used in the process of converting range-ray data to XYZ in the sensor frame. However, it might also be used in rare cases of reversion calculations. In either case, the pulse index (the same as the IPP number) must be known to look up the direction case number. If the lidar is operated in a mode where all pulses use the same ray direction map for the focal plane detector array, this data item would not be populated, and there would be only a single layer of direction cosines in SC.Array.RayDirs.*Ia*(icase,ie,j), that is, the range of icase would be from 1 to 1. When there is enough variation in the element look directions to warrant more than one ray direction map of direction cosine pairs, icase will need to take on a range of values. The specific value to use for a pulse, ip, is the value of SC.Array.DirCaseNum.*TileNum*(ip,j). To find it, one consults the tile map data item PD.TileMap.PulseRange(j,k) to determine the tile in which to find data for the desired pulse. One then evaluates *TileNum* to this value and looks up the needed value of icase in SC.Array.DirCaseNum.*TileNum*(ip,j).
Governing quality metrics	Values are correct/not correct
See Also	PD.TileMap.PulseRange(j,k)

SC.Artifacts.*

SC.Artifacts.* parameters describe the Sensor's known artifacts to support easily determining the likelihood of needed processing strategies to correct

20.5.16 SC.Artifacts.List(j) [L1,2,3,4,5]

Brief Definition	SC.Artifacts.List(j) declares known issues in the collected data
indices	Index, j, ranges from PD.LoIndx.Artifacts to PD.HiIndx.Artifacts, that is over the known list of data issues
Units/representation	Dimensionless, UInt8
Treatment/usage	Each value of j in SC.Artifacts.List(j) matches up with the same value in SC.Artifacts.Descr(j). SC.Artifacts.List(j)=0 if the problem is not present and SC.Artifacts.List(j)=1 if the problem is present. The problem in SC.Artifacts.List(j) is described in SC.Artifacts.Descr(j). A registered list should (eventually) be maintained such that using software need not parse SC.Artifacts.Descr(j) to understand the artifact. This would permit disclosure of only those artifacts present in a data set and only values where the currently defined SC.Artifacts.List(j)=1 would be used, if at all. The registration regime would mean that only SC.Artifacts.ShortList(j)[204] would be populated.[205] It would reference just the registered artifacts that occur in the sensor data. The CMMD currently defines both SC.Artifacts.List(j) and SC.Artifacts.Descr(j) to support full disclosure during the epoch that precedes registration of raw data artifacts.
Governing quality metrics	
See Also	PD.HiIndx.Artifacts SC.Artifacts.Descr(j)

20.5.17 SC.Artifacts.Descr(j) [L1,2,3,4,5]

Brief Definition	SC.Artifacts.Descr(j) describes known issues in the collected data
indices	Index, j, ranges from PD.LoIndx.Artifacts to PD.HiIndx.Artifacts, that is over the known list of data issues
Units/representation	Dimensionless, BCS Character string, length 72
Treatment/usage	This item is a verbal, ideally registered, description of a known data artifact. Informing processors of this enables immediate adaptive responses without having to implement an exhaustive data artifact detection process that increases computational burdens to discover already known facts.
Governing quality metrics	
See Also	SC.Artifacts.List(j)

[204] Addition of this item to the CMMD is deferred until registration of artifacts is implemented.

[205] The intent here is to enable artifact documentation in advance of implementation of a national registry of raw data artifacts. When the artifact registry is operating, we do not recommend populating SC.Artifacts.Descr(j) unless the network services are unable to provide the descriptive text without noticeable impact to users.

SC.Channlzn.*

SC.Channlzn.* parameters describe the Sensor's lidar channelization. This section details the *collection* of raw lidar channelized data. There is a complementary set of channelization parameters (PD.Channlzn.*) that deal with how the channels have been treated in *processing*.

20.5.18 SC.Channlzn.Descr(j,k) [L1,2,3,4,5]

Brief Definition	SC.Channlzn.Descr(j,k) associates a lidar measurement functionality with a number associated with an IPP based data stream of measurements. This describes the collection channelization, even in L2 and above data.
indices	Index, j, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. k=1,2,3,4
Units/representation	Dimensionless, UInt8
Treatment/usage	Per the discussion of the Channel_Model, we associate raw wideband lidar streams with a channel, which is effectively a laser-array/filter pairing. Thus, the key element of identifying a channel is not the hardware data handling, but rather the physical phenomenology represented by the samples in the data. When the sensing array (even if a single "pixel" array) choice alone determines the receiver polar and spectral response, an array and the laser used to illuminate the scene for its samples simply determine the channel and the data for all IPPs constitute the channel data. Only the use of filter wheels cause a trickiness in this definition since the channel is phenomenological but not separate physically. Rather, a single array's output is time multiplexed with channel data depending on wheel position. SC.Channlzn.Descr(j,1)—code for array number SC.Channlzn.Descr(j,2)—code for laser number This is a value "il" the index of the laser illuminators in the laser data. Consistency of index referencing is necessary for this to work. SC.Channlzn.Descr(j,3)—code for spectral filter number[206] SC.Channlzn.Descr(j,4)—code for polarization filter number[206]

[206] These "filter wheel" options may not exist for some sensor designs. The value of the code for "no such filter wheel" is 0.

	The codes used in the sensor configuration (SC) channel description must align with the properties documented in SC.Array.FilterDat(ifc,j) and SC.Array.PolDat(ipc,j).
Governing quality metrics	
See Also	PF.Channlzn.Descr(j) PD.LoIndx.Channlzn.Rg_m PD.HiIndx.Channlzn.ChRg(m) SC.Array.FilterDat(ifc,j) SC.Array.PolDat(ipc,j)

20.5.19 SC.Channlzn.Disp(j) [L1,2,3,4,5]

Brief Definition	SC.Channlzn.Disp(j) documents the intended disposition of the collected channels regardless of whether the data set is L1 or L2.
indices	Index, j, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set.
Units/representation	Dimensionless, UInt8
Treatment/usage	SC.Channlzn.Disp(j) =1 indicates the collection channel is to collect a specific element of the polarization scattering matrix, such as in-plane-of-incidence E-Vector for both the illuminating and detected photons. The channel would then be used to act as part of the polarization scattering matrix and can be processed in terms of transmit and received Stokes vector elements, for example. SC.Channlzn.Disp(j) =2 indicates the collection channel is to collect a specific elastic scattering spectral band. The channel would subsequently be used to create a signature response indicating the presence or absence of some material substance. Pattern matching ala DIAL would then be the exploitation methodology. SC.Channlzn.Disp(j) =3 indicates the collection channel is to collect a specific non-elastic scattering spectral band. The channel would subsequently be used to create a signature response indicating the presence or absence of some material substance. Pattern matching ala Raman spectrometry or analysis of fluorescence would then be the exploitation methodology. SC.Channlzn.Disp(j) =4 indicates the collection channel is to collect a specific (sub)-field of view with respect to the sensor's optical axis. This might be a method of supporting high area rate of collection modes, or of supporting multiple nadir angle collections within a single collection operation.
Governing quality metrics	
See Also	PF.Channlzn.Disp(j)

SC.Laser.*

SC.Laser.* parameters describe sensor configuration data that describe the laser illuminator(s) in the lidar device. There is one laser per channel[207], and the laser description data are described here.

20.5.20 SC.Laser.Dat(il,j) [L1,2,3,4,5]

Brief Definition	SC.Laser.Dat(il,j) supplies key information about the laser illuminators in the lidar sensor that collected the current data set.
indices	Index, il, ranges from PD.LoIndex.Laser(1) to PD.HiIndx.Laser(1), that is over the laser illuminators in the lidar used for this collection. Index, j, ranges from PD.LoIndex.Laser(2) to PD.HiIndx.Laser(2), that is over the documented properties of the laser illuminators in the lidar used for this collection
Units/representation	Various, Float single precision
Treatment/usage	Each of the one or several channels of a lidar sensor will utilize a laser illuminator. That laser illuminator will have some important properties to be documented. SC.Laser.Dat(il,1)—spectral peak wavelength (meters) SC.Laser.Dat(il,2)-- Tx bandwidth parm 1 [λ_{long}] at 1/e power point[208] (meters) SC.Laser.Dat(il,3)-- Tx bandwidth parm 2 [λ_{short}] at 1/e power point (meters) SC.Laser.Dat(il,4)—Expected value of pulse energy (joules) SC.Laser.Dat(il,5)—Standard deviation of pulse energy (joules) SC.Laser.Dat(il,6)—resonant cavity Exit diameter (meters) SC.Laser.Dat(il,7)—Expected pulse duration (to -60 dB points) (seconds) SC.Laser.Dat(il,8)—Expected pulse rise time (start to peak) (seconds) SC.Laser.Dat(il,9)—Time step size in power profile vector (seconds)[209] SC.Laser.Dat(il,10)— transmitted Stokes vector S0 component just after

[207] More specifically, there is one laser type per channel. We treat a gang of like emitters used to illuminate for a single channel as a single laser for purposes of this discussion. Such a group of devices is merely an implementation intended to act as a single laser in lidar operations. Consider the illuminator for a lidar channel to be a "virtual" laser.

[208] This reflects the common modeling of laser lines as Gaussian power functions of wavelength shift from line peak.

[209] It is recommended that this be no greater than the step due to the the Rx sample rate determined from the rate factor, SC.Swath.RateFactor(ia). If not all of the arrays sample laser "il", at the same rate, SC.Laser.Dat(il,9) should be consistent with the highest sampling rate among the arrays that record laser "il".

emerging from the lidar exit aperture
SC.Laser.Dat(il,11)— transmitted Stokes vector S1 component just after emerging from the lidar exit aperture
SC.Laser.Dat(il,12)— transmitted Stokes vector S2 component just after emerging from the lidar exit aperture
SC.Laser.Dat(il,13)— transmitted Stokes vector S3 component just after emerging from the lidar exit aperture
SC.Laser.Dat(il,14)— angle of the major axis of the transmitted polarization ellipse consistent to within 0.1 degree of the orientation determined by the Stokes vector components in SC.Laser.Dat(il,10 through 13). This is measured in degrees counter clockwise looking along the optical axis toward the sensor. The zero degree orientation is that of the direction of SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) for "ia"= PD.LoIndx.Array(1). Notice this is referenced to the receive array orientation vector as given in ECEF by SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) rather than the beam major axis. This is to simplify keeping polarization orientations consistent.

Note that the power profile over the duration of the pulse is provided in SC.WF.TxProfile.Dat(il,j).
Values of il used here must be consistently referenced in channel definition data fields.

Governing quality metrics

See Also: SC.WF.TxProfile.Dat(il,j) PD.HiIndx.Laser(2) PD.LoIndx.Array(1)

20.5.21 SC.Laser.IDat(il,j) [L1,2,3,4,5]

Brief Definition: SC.Laser.IDat(il,j) supplies other key information about the laser illuminators in the lidar sensor that collected the current data set as integer codes.

indices: Index, il, ranges from PD.LoIndex.Laser(1) to PD.HiIndx.Laser(1), that is over the laser illuminators in the lidar used for this collection.
Index, j, ranges from PD.LoIndex.Laser(3) to PD.HiIndx.Laser(3), that is over the documented properties of the laser illuminators in the lidar used for this collection that can be represented by integer codes.

Units/representation: Various, Int32

Treatment/usage: Each of the one or several channels of a lidar sensor will utilize a laser illuminator. That laser illuminator will have some important properties to be documented.
SC.Laser.IDat(il,1)— Material and Doping code

1 ➔ Nd:YAG
2 ➔ Cr:ZnSe

3 ➔ Nd:YVO_4
4 ➔ Nd:SFAP
5 ➔ Nd:YOS
6 ➔ Nd:SVAP
7 ➔Yb:YAG
8 ➔Yb:YVO_4
9 ➔Yb:$KGd(WO_4)_2$ (Yb:KGW)
10 ➔Yb:$KY(WO_4)_2$ (Yb:KYW)
11 ➔Yb:$KLu(WO_4)_2$ (Yb:KLuW)
12 ➔Yb^{3+}:$NaGd(WO_4)_2$ (Yb:NGW)
13 ➔Yb^{3+}:$NaY(WO_4)_2$ (Yb:NYW):
14 ➔Yb:$Sr_3Y(BO_3)_3$ (Yb:BOYS)
15 ➔Yb:$GdCa_4O(BO_3)_3$ (Yb:GdCOB)
16 ➔Yb:$Sr_5(PO_4)_3F$ = Yb:S-FAP
17 ➔Yb:$SrY_4(SiO_4)_3O$ = Yb:SYS
18 ➔Yb:Y_2O_3,
19 ➔Yb:Sc_2O_3
20 ➔Yb:Lu_2O_3
21 ➔Yb:Y_2SiO_5 (Yb:YSO)
22 ➔Yb:Lu_2SiO_5 (Yb:LSO)
23 ➔Yb:Gd_2SiO_5 (Yb:GSO)
24 ➔Yb:$CaGdAlO_4$ (Yb:$CaAlGdO_4$, Yb:CALGO)
25 ➔Yb:CaF_2
26 ➔Yb:SrF_2
27 ➔Tm:$KGd(WO_4)_2$
28 ➔ **QX/Er**
…

SC.Laser.IDat(il,2)— Manufacturer
0 ➔ Custom built by Sensor vendor
1 ➔ Alabama Laser
2 ➔ Coherent
3➔ Elforlight
4➔ SpectraPhysics
5 ➔Laser Energetics
6 ➔Quantronix
…

SC.Laser.IDat(il,3)— Age as percent of rated life at time of collection
0 ➔ Less than 10%
1 ➔ 10 to 50%
2 ➔ 50 to 75%
3 ➔ 75 to 90%

4 ➔ 90 to 100%
5 ➔ 100 to 200%
6 ➔ 200 to 500%
7 ➔ 500 to 1000%
8 ➔ 1000 to 5000%
9 ➔ over 5000%

SC.Laser.IDat(il,4)— Number of command clock counts after the start of collection that the command to fire for the IPP zero of the channel for this laser is issued. The first channel to fire should have this value set to 0.

SC.Laser.IDat(il,5) — Flag indicating whether laser "il" has pulse to pulse transmit power fluctuations tracked in this data set

0 →Laser(il) pulse to pulse Tx power fluctuations are not tracked in this data set

1 →Laser(il) pulse to pulse Tx power fluctuations are tracked in this data set. Refer to SC.WF.TxProfile.Flags(4) to determine whether the fluctuation are tracked in the element sensitivity data with ambiguity number adjustment or are separately documented in the deferred item, SC.Pulse.TxPeakTracker(il,ip), without ambiguity number shift.

SC.Laser.IDat(il,6) — This number is the sample count into the transmit wave form for laser, il, (provided in SC.WF.TxProfile.Dat(il,j)) that indicates the mean time into the waveform that corresponds to the recorded transmit time. This "transmit trigger time mark" information is necessary in order to identify the actual range in point clouds that have a range PSF in evidence (seen as a thickening, or even profiling of the signatures for opaque flat surfaces.) This information could also be applied to analysis of return waveforms in systems that can record the laser returns in this form.

SC.Laser.IDat(il,7) — This number is the sample count before or after the value in SC.Laser.IDat(il,6) to account for 90% of the actual recorded transmit time "marks". Presumably, natural fluctuations in the transmit waveform power profile from pulse to pulse result in some variation in the trigger point for measuring the transmit time. This information is necessary in order to propagate ranging error. So the transmit time recorded is SC.Laser.IDat(il,6) ± SC.Laser.IDat(il,7) P90% in recorded profile time step units.

Values of il used here must be consistently referenced in channel definition data fields.

Governing quality

metrics
See Also PD.HiIndx.Laser(3)

20.5.22 SC.Laser.Beam(il, i, j) [L1,2]

Brief Definition SC.Laser.Beam(il,i,j) is the normalized illuminating beam radiant intensity sampled on a direction cosine grid.

indices Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of lasers. If the lidar design employs more than one laser diffuser or transmitting 'optics chain'/'optics chain setting' and hence there is more than one beam pattern for each laser, the metadata must be populated as if each laser/Tx optics parameter set were, in fact, a separate laser with its specified far field beam pattern. That would require repeating the laser's data (see SC.Laser.Dat, SC.WF.TxProfile.Dat) for each beam pattern option.

Index, i, counts steps of delta cosα, the row to row increment and ranges from PD.LoIndx.Swath(2) to PD.HiIndx.Swath(2)
Index, j, counts steps of delta cosβ, the column to column increment and ranges from PD.LoIndx.Swath(3) to PD.HiIndx.Swath(3)

Units/representation Dimensionless, float

Treatment/usage The beam model is discussed in Appendix 10.1.2.1. The radiant intensity of the beam is the power per unit solid angle and is a function of direction within the beam. The normalized radiant intensity is the radiant intensity at each direction divided by the radiant intensity at the peak of the beam. SC.Laser.Beam(il,i,j) is the normalized illuminating beam radiant intensity sampled on a direction cosine grid. This item may be applied as the deshading function for the illumination pattern at the scene. It is valid to use SC.Laser.Beam(il,i,j) as the radiance arriving at the scene reflector distribution under the condition that propagation effects are uniform across the scene and do not effectively create shadows which are cast on the surface of the scene.

The actual peak (i.e., on-axis) beam intensity in physical units is not provided as a metadata item in this data model because the beam intensity varies in fast time with the pulse transmit power profile. At best, a beam peak at the moment of peak power could be provided. But since the feasibility of a'priori absolute radiometric calibration of lidars used for long range radiometry through the atmosphere is extremenly low due to inherently variable propagation environment uncertainties, we take a different approach in this CMMD. This approach is meant to best support the feasible relative calibration task for spatial rendering of the scene's geometry and approximate

reflectivity and support the generation of confidence metrics about scene rendering quality both geometrically and radiometrically. The beam spatial radiance pattern is given normalized to the beam peak and is an average over many pulses and any local speckle cells due to the transmitter optics. To obtain a value for the peak beam radiance at the exit aperture in physical units, one can combine the beam pattern data in SC.Laser.Beam(il,i,j) with the transmit power profile peak information in SC.WF.TxProfile.Peak(il). SC.WF.TxProfile.Peak(il) scales the fast time transmitted pulse envelope data which are, like the beam pattern, normalized to some peak. If the transmitted pulse is measured at the exit aperture, the photo-current (amps) output from the measuring device is the physical unit directly measured, but the photocurrent is modeled to be linearly proportional to transmitted power (integrated over the observed beam). That measured photo-current will be proportional to the (temporal and spatial) mean exit aperture peak radiance under assumptions of stationary statistics. Note that when determining a fast time peak value of the transmitted beam power in physical units ,we do not attribute the power to a particular solid angle to provide intensity (brightness/ radiance). We perform an average over the entire beam (out to, say, the -60dB contour). To get the value for only the on-axis intensity in physical units, we need to define the solid angle for the optical axis and then use the beam data to figure out what fraction of the total beam power is in the on-axis direction. Also working against providing a value for the peak beam intensity is its variation from pulse to pulse. The above process does not account for the variability of transmitted pulse power in a way that the power on axis for a specific pulse can be determined. The logic in this paragraph is that since we can't generally use the actual intensity level (even if we knew it on a pulse basis) to remotely determine scatterer reflectivity, providing it is a dead end, but one that has a lot of overhead cost. Pulse to pulse channel sensitivity fluctuation at the array element level is available to make the processed sensor response uniform spatially and temporally[210]. See SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k).

Using this item for Geiger mode operation is tricky since the measurements are not proportional to the illumination. It can act as a scaling factor since it is associated with a direction and raw Geiger mode data is also linked to a direction. However, point reflectivity is derived from data accumulation in Geiger mode, so multiple directions play into the intensity estimate for any point. This data item can be valuable for processing and exploitation of linear mode and pulse compression lidars.

[210] On time scales for which the average transmitted power is stationary.

The grid on which the normalized intensity is supplied is described as follows:
Row step size: In-row steps (that is, from column to column) are increments of the direction cosine measured from the row direction axis (see row orientation below). The size of these increments is given by SC.Laser.DelCosAlpha. See figure 10.1.2-2.
Column step size: In-column steps (that is, from row to row) are increments of the direction cosine measured from the column direction axis (see column orientation below). The size of these increments is given by SC.Laser.DelCosBeta. The granularity of the provided beam grid should be fine enough for the sensor to meet its requirements on uniformity of product intensity level.

Row orientation: Provided as a function of pulse number as an ECEF unit vector in SC.Pulse.BmMajorAxis.*TileNum*(j, ip). Per Appendix 10.1.2.1, the beam axis (direction of the beam peak) is normal to the plane of the row and column orientations vectors.
Column orientation: Provided as a function of pulse number as an ECEF unit vector in SC.Pulse.BmMinorAxis.*TileNum*(j, ip)

Location of beam peak in the beam grid: The beam axis[211] is in the direction ($\cos\alpha$, $\cos\beta$)=(0,0). In terms of the indices (i,j), this will appear at the mid value of i and the mid value of j in the ranges described by the values of the Lo and Hi Index values. If the number of rows and number of columns are both odd, the peak will be at a sample point and the value there will be 1.0E00. The enterprise level-1 writer is free to choose the i and j ranges (which it must populate into PD.LoIndx.Swath(2), PD.HiIndx.Swath(2), PD.LoIndx.Swath(3), PD.HiIndx.Swath(3)), but a convenient choice is to run i from (1-Nrows)/2 to (Nrows-1)/2 with Nrows chosen odd so that the middle index is i=0. Similarly it is convenient to run the j index from (1-Ncols)/2 to (Ncols-1)/2 with Ncols chosen odd so that the middle index is j=0. In this way, for example, $\cos\alpha_i$= SC.Swath.DelCosAlpha*i and $\cos\beta_i$=SC.Swath.DelCosBeta*j.

Governing quality metrics

See Also PD.HiIndx.Swath SC.Pulse.BmMajorAxis.*TileNum*(j, ip)

[211] The convention will be that the beam axis is at the peak. The beam data need not be symmetric. If the instrument operators do not maintain alignment of the optical axis of the Tx beam direction control mechanism with the beam peak direction physically, the bias should be compensated when populating the beam scan data at L-1 and L-2.

SC.Laser.DelCosBeta PD.HiIndx.Laser

20.5.23 SC.Laser.DelCosAlpha [L1,2]

Brief Definition	SC.Laser.DelCosAlpha is the beam data grid direction cosine step size for along row increments, which lie along the beam major axis orientation.
Indices	none[212]
Units/representation	Dimensionless, float
Treatment/usage	SC.Laser.DelCosAlpha is a support data item definining the granularity of the sampling of the beam data grid. SC.Laser.DelCosAlpha is the increment of a uniform grid in direction cosine space based on the cosine of the angle between a direction vector within the beam and the beam major axis direction vector as supplied by SC.Pulse.BmMajorAxis.*TileNum*(j, ip). Per figure 10.1.2-2, α is an angle between the major axis direction and the beam sample direction, so cosα corresponds steps along the row direction, not between rows. Note that steps along the row direction are column to column separations.
Governing quality metrics	Compliance to definition. Accuracy (true step size-populated step size)/true step size
See Also	SC.Pulse.BmMajorAxis.*TileNum*(j, ip) SC.Laser.Beam(il,i,j)

20.5.24 SC.Laser.DelCosBeta[L1,2]

Brief Definition	SC.Laser.DelCosBeta is the beam data grid direction cosine step size for along column increments, which lie along the beam minor axis orientation.
indices	none
Units/representation	Dimensionless, float
Treatment/usage	SC.LAser.DelCosBeta is a support data item defining the granularity of the sampling of the beam data grid. SC.Laser.DelCosBeta is the increment of a uniform grid in direction cosine space based on the cosine of the angle between a direction vector within the beam and the beam minor axis direction vector as supplied by SC.Pulse.BmMinorAxis.*TileNum*(j, ip).

[212] Advanced designs may zoom control the beam to keep a constant beam footprint size on the ground. Such beam variation would modify the beam table direction cosine steps within a scan group and phase dependence to permit the sensor to spread the beam as needed with instantaneous (slow time) collection geometry. The current data item need not be changed when such systems emerge. A zoom factor for DelCosAlpha could be later added as a new metadata parameter item and it could be a function of scan phase for each scan cone. The new metadata items would have in their definition directions to treat SC.Laser.DelCosAlpha (and beta) as the value at the reference value of the scan phase.

	Per figure 10.1.2-2, β is an angle between the minor axis direction and the beam sample direction, so cosβ corresponds steps along the column direction, not between columns. Note that steps along the column direction are row to row separations.
Governing quality metrics	Compliance to definition. Accuracy (true step size-populated step size)/true step size
See Also	SC.Pulse.BmMinorAxis.*TileNum*(j, ip) SC.Laser.Beam(il,i,j)

SC.MultiDwell.*

SC.MultiDwell.* parameters describe sensor configuration/mode design data that describe the status of the data with respect to multi-dwell strategy. This relieves the need to perform a complete scan parameters analysis to determine whether disjoint time intervals have been devoted to single contiguous voxel regions.

20.5.25 SC.MultiDwell.Dat(j) [L1,2,3,4,5]

Brief Definition	SC.MultiDwell.Dat(j) provides summary parameters for multi-dwell data sets at levels below 4.
indices	Index, j, ranges from PD.LoIndx.Swath(10) to PD.HiIndx.Swath(10), that is over the parameters available to describe the dwells in a multi-dwell collection.
Units/representation	Dimensionless, UInt16
Treatment/usage	SC.MultiDwell.Dat(j) should be populated for SC.MultiDwell.Flg=1 cases and does not apply for SC.MultiDwell.Flg=0. SC.MultiDwell.Dat(1)= Number of IPPs in a dwell SC.MultiDwell.Dat(2)= Number of IPPs between dwells SC.MultiDwell.Dat(3)= Number of Dwells for each scene point
Governing quality metrics	
See Also	PD.HiIndx.Swath(10)

20.5.26 SC.MultiDwell.Flg[L1,2,3,4,5]

Brief Definition	SC.MultiDwell.Flg is a code indicating the status of the data set with respect to multi-dwell collection strategy.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	SC.MultiDwell.Flg=0, also the default if the data is not populated, indicates that contiguous voxel regions of the scene have been subject to only a single dwell.

	SC.MultiDwell.Flg=1, indicates that contiguous voxel regions of the scene have been subject to multiple disjoint dwells.
Governing quality metrics	
See Also	

SC.Overlap.*

SC.Overlap.* parameters describe sensor operation based field of view overlaps in pulse to pulse, scan to scan and subswath to subswath coverage. This data supports

1) stage to stage processing so if data is tapped, the ingesting application can see where registrations might be possible, and assess that against what has happened so far in each overlap region.
2) Error analysis applications can see where overlaps in collection exist to support arriving at error estimates.
3) Assessment of multiple interrogations of scene

20.5.27 SC.Overlap.PulsePrcnt[L1,2]

Brief Definition	SC.Overlap.PulsePrcnt documents pulse to pulse overlap of collected instantaneous FOV in support of "frame to frame" registration processing, if it is employed.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	This parameter is necessary in the case when intra-scan, pulse to pulse registration is needed to meet system geometric distortion requirements. It is assumed that only consecutive pulses worth of data (sometimes referred to as frames) would ever be registered. The nominal ("error free") overlap range is provided in SC.Overlap.Pulse.X_Lo.TileNum(ip) etc. Even if the system design does not incorporate "frame to frame" adaptive registration, this data supports evaluation of the number of times a region is interrogated by the sensor. SC.Overlap.PulsePrcnt gives the nominal (integer) percentage of area of each array FOV in an overlap region. So two adjacent FOVs from consecutive pulses with area of 100 sq units each, having 2 square units in common with its overlap mate have 2 percent[213] overlap. This is a convenient L2 data item.

[213] Zero overlap gets a value of 0 here. For overlap percentage values greater than 0 and less than 99 percent, we will round up to a reported range of 1 to 99percent. For overlaps predicted to be greater than or equal to 99 and less than 100 percent, we will round down to 99 so that only 100 % intended overlap gets identified as 100%. This value

Governing quality metrics	
See Also	SC.Overlap.Pulse.X_Lo.TileNum(ip) SC.Overlap.Pulse.X_Hi.TileNum(ip) SC.Overlap.Pulse.Y_Lo.TileNum(ip) SC.Overlap.Pulse.Y_HiTileNum(ip)

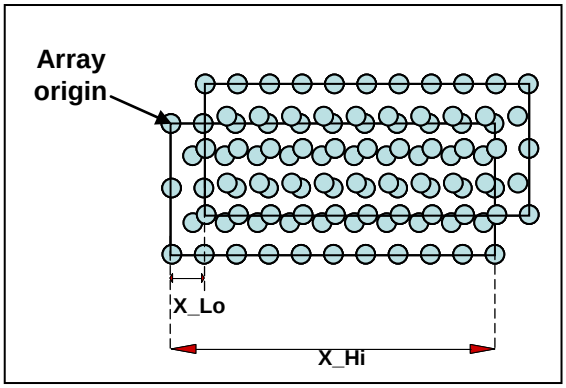
20.5.28 SC.Overlap.Pulse.X_Hi(j,k) [L1,2]

Brief Definition	SC.Overlap.Pulse.X_Hi(j,k) gives the nominal along row upper bound of coverage overlap for consecutive frames.
indices	Index, j, ranges from PD.LoIndx.Overlap(1) to PD HiIndx.Overlap(1) , that is, over the pulse pairs in a single scan Index, k, ranges from PD.LoIndx.Overlap(2) to PD.HiIndx.Overlap(2) , that is, over the scans in a single collection subswath
Units/representation	Array row direction steps, Float
Treatment/usage	The presence of overlap between successive projections of the focal plane detector array to the ground/scene[214] not only leads to multiple interrogations of scene points, but presents the opportunity for adaptive registration of successive "frames" when that is felt to improve geometric integrity of the data. To support assessment of these two issues, a processor/user needs to know the nominal [that is, the intended value in the absence of system error] value of the array footprint's overlap in consecutive pulses. This value is not necessarily a constant and is thus provided on a pulse basis throughout the collection of a subswath. [215]

would never be used, since it prevents a scan from developing. This value is used as a seed in a search process and is always nominal since the planned overlap is all that can go here. Actual overlap has to be determined by analysis of the collection data.

[214] This presumes a plane of primary interest in the scene at some range is used to compute overlaps. A downward looking assumption is common, but not necessary for this data definition.

[215] Need to add that this definition assumes that all collection subswaths are equivalent, thus no index for which subswath is being described. Also, we characterize overlap as if there is no rotation at the pulse to pulse level. This is because scans are presumed to have only along and across array planned shifts. Real registration information must be adaptively determined and can't be given at the SC.* level.



The figure shows the footprint of the Focal plane array as projected to the ground in two consecutive pulses, N and N+1. The origin of the projected array frame is indicated for pulse N. There is a common overlapping region of coverage and the boundary of the highest distance measured along a row from the origin for pulse N is at SC.Overlap.Pulse.X_Hi(N,k) for the kth scan in a subswath.
It is not permitted for the value to be negative or greater than the row extent of the array, as this would represent a position not on the array for pulse N.

If the scan control of the sensor is operated with angle space constraints, the number of pulses in a scan could actually vary slightly. This could arise for constant PRF when the along track off-nadir offset gets significant. If the angular rate of the steering mechanism varies, the number of pulses per scan could change within a subswath. The table [(j,k) matrices for low and high bounds in the row and cross row directions] would carry the max number of pulses across the scans in a subswath. Coding of negative values for overlap bounds could signal that the data in the cell is intended to be ignored.

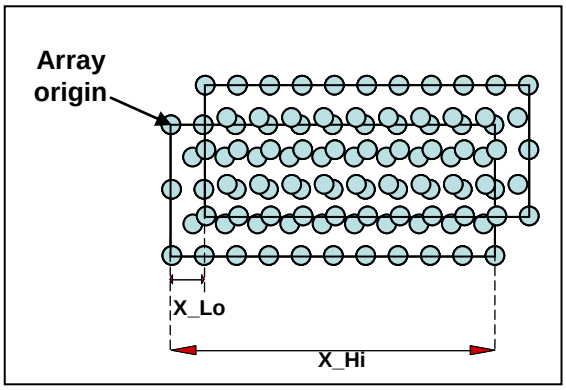
When the subswaths are the consequence of a sensor configuration, each subswath is a repeat of the same set of scan articulations relative to the sensor. Thus, all collection subswaths use the subswath data in SC.Overlap.Pulse.X_Hi(j,k).

Governing quality metrics: Percentage error in supplied value

See Also: SC.Overlap.Pulse.Y_Lo(j,k) SC.Overlap.Pulse.X_Lo(j,k) PD.HiIndx.Overlap(j)

20.5.29 SC.Overlap.Pulse.X_Lo(j,k) [L1,2]

Brief Definition	SC.Overlap.Pulse.X_Lo(j,k) gives the nominal along row lower bound of coverage overlap for consecutive pulses.
indices	Index, j, ranges from PD.LoIndx.Overlap(1) to PD HiIndx.Overlap(1) , that is, over the pulse pairs in a single scan Index, k, ranges from PD.LoIndx.Overlap(2) to PD.HiIndx.Overlap(2) , that is, over the scans in a single collection subswath
Units/representation	Float, Array row direction steps
Treatment/usage	The presence of overlap between successive projections of the focal plane detector array to the ground/scene[214] not only leads to multiple interrogations of scene points, but presents the opportunity for adaptive registration of successive "frames" when that is felt to improve geometric integrity of the data. To support assessment of these two issues, a processor/user needs to know the nominal [that is, the intended value in the absence of system error] value of the array footprint's overlap in consecutive pulses. This value is not necessarily a constant and is thus provided on a pulse basis throughout the collection of a subswath.



The figure shows the footprint of the Focal plane array as projected to the ground in two consecutive pulses, N and N+1. The origin of the projected array frame is indicated for pulse N. There is a common overlapping region of coverage and the boundary of the lowest distance measured along a row from the origin for pulse N is at SC.Overlap.Pulse.X_Lo(N,k) for the kth scan in a subswath.

It is not permitted for the value to be negative or greater than the row extent of the array, as this would represent a position not on the array for pulse N.

Governing quality metrics	Percentage error in supplied value
See Also	SC.Overlap.Pulse.Y_Lo(j,k) SC.Overlap.Pulse.X_Lo(j,k) PD.HiIndx.Overlap(j)

20.5.30 SC.Overlap.Pulse.Y_Hi(j,k) [L1,2]

Brief Definition

SC.Overlap.Pulse.Y_Hi(j,k) gives the nominal cross-row upper bound of coverage overlap for consecutive pulses.

indices

Index, j, ranges from PD.LoIndx.Overlap(1) to PD HiIndx.Overlap(1) , that is, over the pulse pairs in a single scan

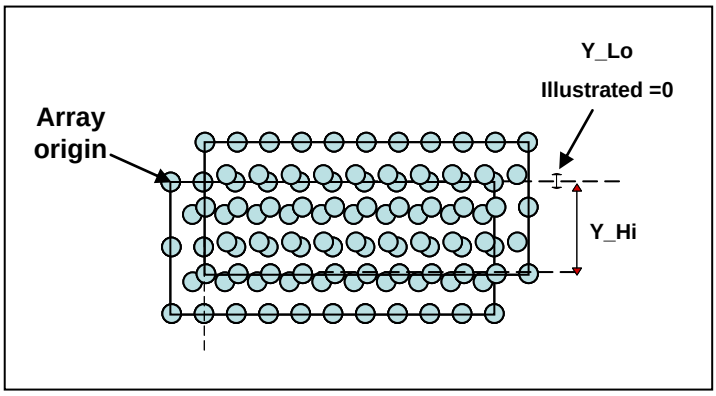
Index, k, ranges from PD.LoIndx.Overlap(2) to PD.HiIndx.Overlap(2) , that is, over the scans in a single collection subswath

Units/representation

Float, Array cross-row direction steps

Treatment/usage

The presence of overlap between successive projections of the focal plane detector array to the ground/scene[214] not only leads to multiple interrogations of scene points, but presents the opportunity for adaptive registration of successive "frames" when that is felt to improve geometric integrity of the data. To support assessment of these two issues, a processor/user needs to know the nominal [that is, the intended value in the absence of system error] value of the array footprint's overlap in consecutive pulses. This value is not necessarily a constant and is thus provided on a pulse basis throughout the collection of a subswath.



The figure shows the footprint of the Focal plane array as projected to the ground in two consecutive pulses, N and N+1. The origin of the projected array frame is indicated for pulse N. There is a common overlapping region of coverage and the boundary of the highest distance measured along a cross-row from the origin for pulse N is at SC.Overlap.Pulse.Y_Hi(N,k) for the kth scan in a subswath.

It is not permitted for the value to be negative or greater than the cross-row extent of the array, as this would represent a position not on the array for pulse N.

Governing quality metrics

See Also

SC.Overlap.Pulse.X_Hi(j,k) SC.Overlap.Pulse.Y_Lo(j,k) PD.HiIndx.Overlap(j)

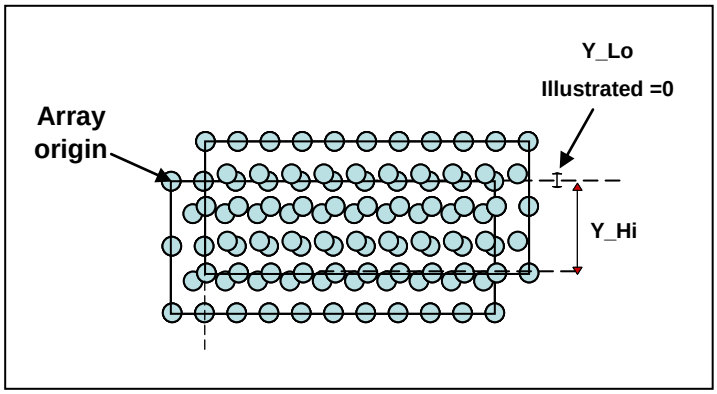
20.5.31 SC.Overlap.Pulse.Y_Lo(j,k) [L1,2]

Brief Definition: SC.Overlap.Pulse.Y_Lo(j,k) gives the nominal cross-row lower bound of coverage overlap for consecutive pulses.

indices: Index, j, ranges from PD.LoIndx.Overlap(1) to PD HiIndx.Overlap(1) , that is, over the pulse pairs in a single scan
Index, k, ranges from PD.LoIndx.Overlap(2) to PD.HiIndx.Overlap(2) , that is, over the scans in a single collection subswath

Units/representation: Float, Array cross-row direction steps

Treatment/usage: The presence of overlap between successive projections of the focal plane detector array to the ground/scene[214] not only leads to multiple interrogations of scene points, but presents the opportunity for adaptive registration of successive "frames" when that is felt to improve geometric integrity of the data. To support assessment of these two issues, a processor/user needs to know the nominal [that is, the intended value in the absence of system error] value of the array footprint's overlap in consecutive pulses. This value is not necessarily a constant and is thus provided on a pulse basis throughout the collection of a subswath.



The figure shows the footprint of the Focal plane array as projected to the ground in two consecutive pulses, N and N+1. The origin of the projected array frame is indicated for pulse N. There is a common overlapping region of coverage and the boundary of the lowest distance measured in the cross-row direction from the origin for pulse N is at SC.Overlap.Pulse.Y_Lo(N,k) for the kth scan in a subswath. In the diagram a special case is illustrated where the advance of the coverage area is in the negative cross row direction. Consequently, the low boundary of the common region is the axis in the row direction through the origin. The analogous case could arise for the X_Lo/Hi bounds as well.
It is not permitted for the value to be negative or greater than the row extent of the array, as this would represent a position not on the array for pulse N.

Governing quality: Percentage error in supplied value

metrics	
See Also	SC.Overlap.Pulse.X_Lo(j,k) SC.Overlap.Pulse.Y_Hi(j,k) PD.HiIndx.Overlap(j)

20.5.32 SC.Overlap.ScanPrcnt [L1,2]

Brief Definition	SC.Overlap.ScanPrcnt documents collected scan to scan overlap to support scan to scan registration, if it is employed.
indices	none
Units/representation	Dimensionless, UInt8
Treatment/usage	SC.Overlap.ScanPrcnt is intended to support intra-(sub[216])-swath scan to scan registration. This parameter is necessary in the case when intra-swath, scan to scan registration is needed to meet system geometric distortion/3-D image quality requirements. It is assumed that only overlapping adjacent scans data would ever be registered. The error free overlap range is provided in SC.Overlap.Scan.X_Lo(j) etc. and the scan indices that constitute the pairs are provided in SC.Overlap.Scan.ID1(j) and SC.Overlap.Scan.ID2(j). SC.Overlap.ScanPrcnt is the nominal (integer) percentage of area of each scan in overlap region. So two adjacent scans with area of 100 sq units each, having 2 square units in common with its overlap mate have 2 percent overlap.
Governing quality metrics	
See Also	SC.Overlap.Scan.ID1(j) SC.Overlap.Scan.ID2(j) SC.Overlap.Scan.X_Lo(j) SC.Overlap.Scan.X_Hi(j)

20.5.33 SC.Overlap.Scan.ID1(j) [L1,2]

Brief Definition	SC.Overlap.Scan.ID1(j) is a list of scan index values, each of which is an identifier for the first scan in a long list of scan pairs. Taken together with SC.Overlap.Scan.ID2(j), a list of scan pairs is realized. The primary processing operation on these pairs is registration.
indices	The index j runs from PD.LoIndx.Overlap(3) to PD.HiIndx.Overlap(3), that is, over the individual scan pairs. A commonly used range would be from 1 to the number of scan pairs to be registered.
Units/representation	Dimensionless, UInt16

[216] The parenthetical (sub) indicates that scan to scan registration may be needed whenever scans are adjacent in the swath. This can occur whether or not there are collection subswaths. When there are collection subswaths, the scan to scan registrations would be used to reduce error in constructing a subswath. Registration of subswaths to each other is not swath to swath registration because subswaths are geographical blocks contained within a single collection operation's coverage.

Treatment/usage

Scans and scan parameterization are discussed in 10.1.4.2 and 10.1.2.4. Per these discussions, the processing application that writes an enterprise level-1 or-2 data set may associate a scan index (notated as "is" in this document) with certain block sequences of pulses in the data collection. These sequences each correspond to a single instance of a repeated articulation pattern of the beam axis direction of the lidar sensor. The value of "is" increments consecutively with each of these pattern completions. The value of "is" is then a label for a particular scan in the collected data set. We colloquially refer to the lidar returns for the pulses in one of these scan operations as scans and a common processing task is to register these scans. Registration is only possible when there is common scene coverage in two scans[217]. So the pairs, [SC.Overlap.Scan.ID1(j), SC.Overlap.Scan.ID2(j)] will be confined to paired values of "is" where at least some of the pulses in each scan were directed to the same earth locations. This is a natural consequence of collection with scan to scan overlap for consecutive scans. It may also occur for non- consecutive scans when the scans are parts of consecutive collection subswaths. Pairs consisting of the values of SC.Overlap.Scan.ID1(j) and SC.Overlap.Scan.ID2(j) for any single value of j then may or may not be consecutive, but they will always have some nominal degree of overlap. That degree is supplied in SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j), SC.Overlap.Scan.Y_Lo(j), SC.Overlap.Scan.Y_Hi(j).
The beginning of the list has any scan-scan pairs (to be) registered in building subswaths. The second part of the list has the pairs for subswath to subswath registration. Thus, specific values of the scan index may occur for more than a single value of j.

Governing quality metrics

See Also

SC.Overlap.Scan.ID2(j) SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) PD.HiIndx.Overlap(j)

20.5.34 SC.Overlap.Scan.ID2(j) [L1,2]

Brief Definition

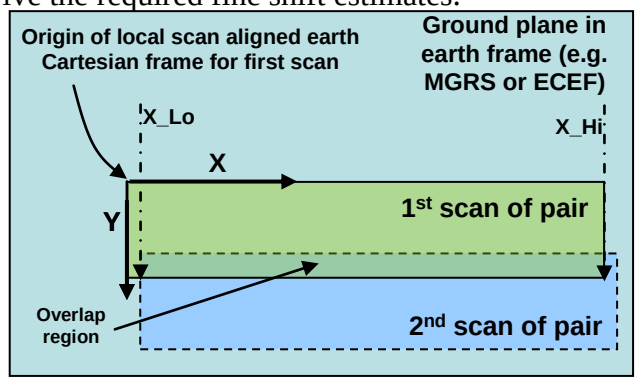
SC.Overlap.Scan.ID2(j) is a list of scan index values, each of which is an identifier for the second scan in a long list of scan pairs. Taken together with SC.Overlap.Scan.ID1(j), a list of scan pairs is realized. The primary processing operation on these pairs is registration.

[217] When features are extrapolated and apparent shears are removed in abutting scans, we should call this shear adjustment, not registration.

indices	The index j runs from PD.LoIndx.Overlap(3) to PD.HiIndx.Overlap(3), that is, over the individual scan pairs. A commonly used range would be from 1 to the number of scan pairs to be[218] registered.
Units/representation	Dimensionless, UInt16
Treatment/usage	The jth scan pair is [SC.Overlap.Scan.ID1(j), SC.Overlap.Scan.ID2(j)]. For information about how the scan index values are assigned, see discussion for SC.Overlap.Scan.ID1(j).
Governing quality metrics	
See Also	SC.Overlap.Scan.ID1(j), SC.Overlap.Scan.Y_Lo(j), SC.Overlap.Scan.Y_Hi(j).

20.5.35 SC.Overlap.Scan.X_Hi(j) [L1,2]

Brief Definition	SC.Overlap.Scan.X_Hi(j) is the nominal X- direction upper bound of coverage overlap for a scan pair.
indices	The index j runs from PD.LoIndx.Overlap(3) to PD.HiIndx.Overlap(3), that is, over the individual scan pairs. A commonly used range would be from 1 to the number of scan pairs to be, or that have been registered.
Units/representation	Meters, Float
Treatment/usage	For registration algorithms to work, they need to be fed with data blocks that are expected to have portions over which there is high correlation, that is, where the scene being rendered is the same. The job of these algorithms is to get a fine estimate for shifts between input data blocks. The job of SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) is to indicate the nominal region in which these algorithms can expect substantial correlation and can derive the required fine shift estimates.



[218] This item may be populated even after L-2 registration, but it documents intended registration, not "in fact" registration.

SC.Overlap.Scan.X_Hi is a coordinate in the ground/scene plane Cartesian frame aligned with the first scan of the pair. SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) give the nominal [assumes absence of system errors] bounds of the region in the first scan overlapped by the second scan. This is normally planned into the collection.

Governing quality metrics

See Also SC.Overlap.Scan.X_Lo(j) SC.Overlap.Scan.Y_Lo(j) SC.Overlap.Scan.Y_Hi(j) PD.HiIndx.Overlap

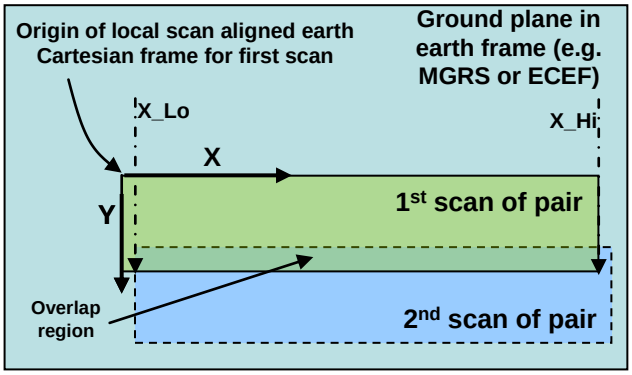
20.5.36 SC.Overlap.Scan.X_Lo(j) [L1,2]

Brief Definition SC.Overlap.Scan.X_Lo(j) is the nominal X- direction lower bound of coverage overlap for a scan pair.

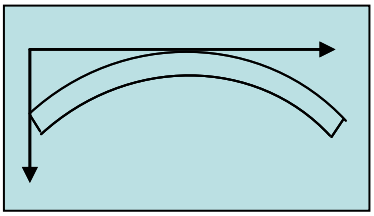
indices The index j runs from PD.LoIndx.Overlap(3) to PD.HiIndx.Overlap(3), that is, over the individual scan pairs. A commonly used range would be from 1 to the number of scan pairs to be registered.

Units/representation Meters, Float

Treatment/usage For registration algorithms to work, they need to be fed with data blocks that are expected to have portions over which there is high correlation, that is, where the scene being rendered is the same. The job of these algorithms is to get a fine estimate for shifts between input data blocks. The job of SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) is to indicate the nominal region in which these algorithms can expect substantial correlation and can derive the required fine shift estimates.



SC.Overlap.Scan.X_Lo is a coordinate in the ground/scene plane Cartesian frame aligned with the first scan of the pair. SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) give the nominal [assumes absence of system errors] bounds of the region in the first scan overlapped by the second scan. This is normally planned into the collection. In the event that the scan projection to the ground is not rectangular, the scan should be interpreted as embedded in the Cartesian frame aligned with tangents to the scan at the mid points of the longest edges and their normals.



If the frame projected to the scanning plane of the ground/scene is a parallelogram, SC.Overlap.Scan.* are referenced to the corresponding deskewed values of Cartesian ground/scene positions associated with the projected rows of detector elements.

Governing quality metrics

See Also SC.Overlap.Scan.X_Hi(j)

20.5.37 SC.Overlap.Scan.Y_Hi(j) [L1,2]

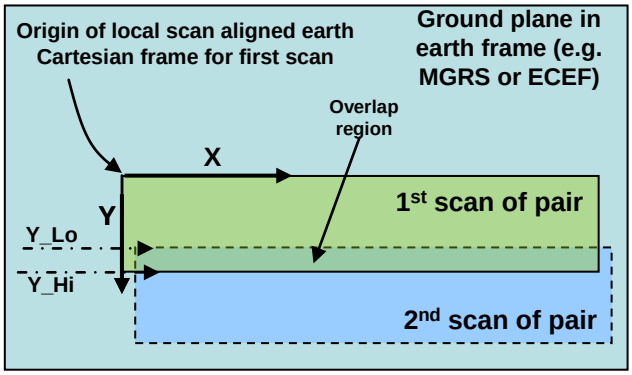
Brief Definition SC.Overlap.Scan.Y_Hi(j) is the nominal Y- direction upper bound of coverage overlap for a scan pair.

indices The index j runs from PD.LoIndx.Overlap(3) to PD.HiIndx.Overlap(3), that is, over the individual scan pairs. A commonly used range would be from 1 to the number of scan pairs to be registered.

Units/representation Meters, Float

Treatment/usage For registration algorithms to work, they need to be fed with data blocks that are expected to have portions over which there is high correlation, that is, where the scene being rendered is the same. The job of these algorithms is to get a fine estimate for shifts between input data blocks. The job of SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) is to indicate the nominal region in which these algorithms can expect substantial correlation

and can derive the required fine shift estimates.



SC.Overlap.Scan.Y_Hi is a coordinate in the ground plane Cartesian frame aligned with the first scan of the pair. SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) give the nominal [assumes absence of system errors]bounds of the region in the first scan overlapped by the second scan. This is normally planned into the collection.

Governing quality metrics

See Also SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.X_Lo(j)

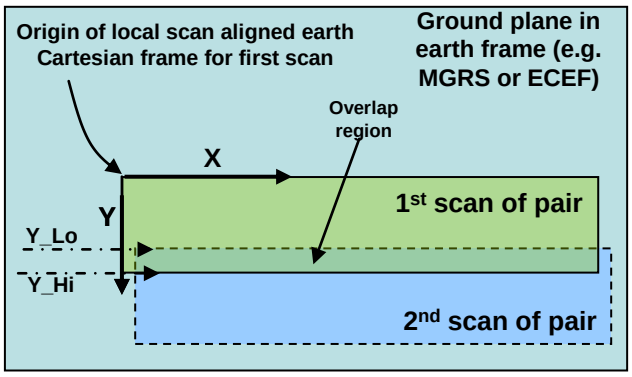
20.5.38 SC.Overlap.Scan.Y_Lo(j) [L1,2]

Brief Definition SC.Overlap.Scan.Y_Lo(j) is the nominal Y- direction lower bound of coverage overlap for a scan pair.

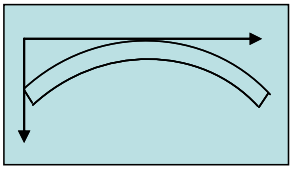
indices The index j runs from PD.LoIndx.Overlap(3) to PD.HiIndx.Overlap(3), that is, over the individual scan pairs. A commonly used range would be from 1 to the number of scan pairs to be, or that have been registered.

Units/representation Meters, UInt8

Treatment/usage For registration algorithms to work, they need to be fed with data blocks that are expected to have portions over which there is high correlation, that is, where the scene being rendered is the same. The job of these algorithms is to get a fine estimate for shifts between input data blocks. The job of SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) is to indicate the nominal region in which these algorithms can expect substantial correlation and can derive the required fine shift estimates.



SC.Overlap.Scan.Y_Lo is a coordinate in the ground plane Cartesian frame aligned with the first scan of the pair. SC.Overlap.Scan.X_Lo(j), SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Lo(j) and SC.Overlap.Scan.Y_Hi(j) give the nominal [assumes absence of system errors] bounds of the region in the first scan overlapped by the second scan. This is normally planned into the collection. In the event that the scan projection to the ground is not rectangular, the scan should be interpreted as embedded in the Cartesian frame aligned with tangents to the scan at the mid points of the longest edges and their normals.



Governing quality metrics

See Also SC.Overlap.Scan.X_Hi(j) SC.Overlap.Scan.Y_Hi(j) SC.Overlap.Scan.X_Lo(j)

20.5.39 SC.Overlap.SubswathPrcnt [L1,2]

Brief Definition SC.Overlap.SubswathPrcnt documents collected subswath to subswath overlap in support of subswath registration.

indices none

Units/representation Dimensionless, UInt8

Treatment/usage

This data applies to collected subswath to subswath overlap. Such overlap will be collected to support registration of blocks of scans to each other and/or provide margin against voids in the face of the realities of collection platform stability challenges. Per 10.1.4.3, a collection may employ sub-sequences of scans that are called subswaths. There will generally be some degree of overlap between the scans of one subswath and the scans of an adjacent subswath. The presence of overlapping data permits registration of one subswath to another using the same approach as that used to assemble a subswath from registered constituent scans.
SC.Overlap.SubswathPrcnt is the nominal (integer) percentage of area of each subswath in the overlap region. So two adjacent subswaths with area of 100 sq units each, having 2 square units in common with its overlap mate have 2 percent overlap. Rounding rules for 0-1% and 99-100%: Only 0 and 100% get called 0 and 100%. So any other values are rounded to 1 or 99 as appropriate.

Governing quality metrics

See Also

SC.Pulse.*

SC.Pulse.* parameters describe sensor configuration items that need to be mapped to individual pulses. This data permits the pulse index for a point to map to those aspects of a point's collection that are due to the pulse index alone so that the data redundancy is reduced by a factor of the number of detectors on an array.

20.5.40 SC.Pulse.ArrayMajorAxis.*TileNum*(ia, ip, j) [L1,2]

Brief Definition

SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) gives the ECEF direction of the major axis of the focal plane detector array for each pulse.

indices

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays.

Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to

PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.

The index, j, ranges from 1 to 3 to cover the X,Y,Z components respectively of a direction vector.

Units/representation Float, Dimensionless

Treatment/usage SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) data supports the computation of the precise direction and orientation of the effective optical axis of the lidar receiver. This data is to be used for computing XYZ point locations in the sensor frame. [The scan parameterization (e.g. evaluating the scan cone at the phase for a given pulse found in SC.Pulse.Scan.Phase.*TileNum*(ip, j)) of the direction of the receiver optical axis is not intended to supplant this data, but rather to be a more convenient and possibly less accurate indicator of the scanning pattern.] As discussed in 10.1.2.2, the effective focal plane detector array orientation is given by two ECEF vectors, SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) and SC.Pulse.ArrayMinorAxis.*TileNum*(ia, ip,j). As long as the array minor axis is in the effective focal plane and is not in the same direction as the array major axis, the line of sight and the orientation of the array about the line of sight are computable in the ECEF frame. The cross product[219] of SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) and SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) gives the direction of the instantaneous steered optical axis (line of sight) of the lidar. The direction of SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) establishes the orientation of the array about the line of sight axis, measured by the angle “gamma”.

A recommended strategy is to associate the direction indicated for each pulse in SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) with the focal plane row direction where this is appropriate. In the case of a single element array, the orientation is arbitrary, but the array major axis direction of array ia= PD.LoIndx.Array(1) will still be used as the basis for each receiver polarization x-direction definition.

[219] It matters not what the order is as long as the direction is taken from sensor to scene after forming the cross product. The model is that if the array grid is viewed as in the 4th quadrant, the minor axis crossed into the major axis (row direction is normal in the Line of sight direction. The array has a row and cross row direction defined in SC.Array.OptclCent(j). However, the row direction could be minor, so we use common sense to determine the LOS, rather than an extra data item.

The calculation of the array major and minor axis directions should include any necessary correction for light aberration due to platform motion necessary to meet system accuracy requirements[220].
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

The beam axes and the array axes are provided as independent data items so that the direction of the illuminating beam optical axis and of the receive optical axis for a given pulse can be adjusted to be different if needed to optimize sensor performance (might enhance performance in the presence of aberration). Additionally, the orientation of the beam and receive array patterns about their respective optical axes can be distinct.

Governing quality metrics

See Also SC.Pulse.BmMajorAxis.TileNum(il,ip,j) 10.1.2.2 ES.Scan.SigmaGamma(j) SC.Pulse.Scan.Phase.*TileNum*(ip, j)

20.5.41 SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) [L1,2]

Brief Definition SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) gives the ECEF direction of the minor axis of the focal plane detector array for each pulse.

indices Index, ia, ranges from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane arrays.

Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
The index, j, ranges from 1 to 3 to cover the X,Y,Z components respectively of a direction vector.
TileNum, though part of a tag name, acts somewhat as an index. It indicates

[220] This puts a burden on the L-1 writer and sensor operator to meet location accuracy requirements via adjustment of the focal plane orientation so that its normal is along the optical axis of light coming from the center of the instantaneous FOV. This is an error budget issue and for low velocity or loose geolocation requirements, this may be a negligible effect.

which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Float, Dimensionless

Treatment/usage: As discussed in 10.1.2.2, the effective focal plane detector array orientation is given by two ECEF vectors, SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) and SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j). As long as the array minor axis is in the effective focal plane and is not in the same direction as the array major axis, the line of sight and the orientation of the array about the line of sight are computable in the ECEF frame. The cross product gives the direction of the instantaneous steered optical axis (line of sight) of the lidar. The direction of SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j) establishes the orientation of the array about the line of sight axis.
A recommend strategy is to associate the direction indicated for each pulse in SC.Pulse.ArrayMinorAxis.*TileNum*(ia,ip,j) with the focal plane cross-row direction where this is appropriate.

The calculation of the array major and minor axis directions should include any correction for light aberration due to platform motion necessary to meet system accuracy requirements.

Governing quality metrics: Value should be correct to the level needed to compute earth frame XYZ to required accuracy in the absence of propagation error.

See Also: SC.Pulse.ArrayMajorAxis.*TileNum*(ia,p,j)

20.5.42 SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j) [L1,2]

Brief Definition: SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j) gives the ECEF direction of the major axis of the transmitted beam for each pulse.

indices: Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of laser beam pattern axis values. If the lidar design employs more than one laser diffuser or transmitting ‘optics chain’/‘optics chain setting’ and hence there is more than one beam pattern aspect or orientation for each laser, the metadata must be populated as if each laser/Tx optics parameter set were in fact a separate laser illuminator with its specified far field beam pattern. That would require counting each (laser,laser optics chain) a value of “il” in PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1) so SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j) can cover each beam pattern option.

Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges

from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.

The index, j, ranges from 1 to 3 to cover the X,Y,Z components respectively of a direction vector.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Float, Dimensionless

Treatment/usage: Define the beam plane to be that of the surface of constant phase[221] at the optical exit aperture. Further, allow that contours of constant power are approximately elliptical. [circular is the degenerate case of elliptical with equal major and minor axes]. Analogously to specifying the detector array geometry by two vectors, the transmit beam is characterized by two vectors. Their cross product determines the beam axis and the major axis determines the orientation of the beam pattern about the beam axis. The direction of the SC.Pulse.BmMajorAxis.*TileNum*(ip,j) is along the far field major axis direction. In the case that the far field beam pattern is circular, it is recommended that the major beam axis be chosen to be parallel to the array major axis for the same channel. The beam axes and the array axes are provided as independent data items so that the direction of the illuminating beam optical axis and of the receive optical axis for a given pulse can be adjusted to be different if needed to optimize sensor performance (might enhance performance in the presence of aberration). Additionally, the orientation of the beam and receive array patterns about their respective optical axes can be distinct.

Governing quality metrics:

See Also: SC.Pulse.ArrayMajorAxis.*TileNum*(ip,j) PD.HiIndx.Laser(j)

[221] If the phase is not actually planar at the exit aperture, then the plane we speak of is the tangent to a parabolic fit (over the part of the beam above -3dB from the peak) to the phase surface at the beam power centroid. For well formed beam patterns, the beam peak will be at the power centroid within the measurement accuracy.

20.5.43 SC.Pulse.BmMinorAxis.*TileNum* (il,ip,j) [L1,2]

Brief Definition	SC.Pulse.BmMinorAxis.*TileNum*(il,ip,j) gives the ECEF direction of the minor axis of the transmitted beam for each pulse.
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of laser beam pattern axis values. If the lidar design employs more than one laser diffuser or transmitting 'optics chain'/'optics chain setting' per laser and hence there is more than one beam pattern aspect or orientation for each laser, the metadata must be populated as if each laser/Tx optics parameter set were in fact a separate laser illuminator with its specified far field beam pattern. That would require counting each (laser,laser optics chain) a value of "il" in PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1) so SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j) can cover each beam pattern option. Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles. The index, j, ranges from 1 to 3 to cover the X,Y,Z components respectively of a direction vector. *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.
Units/representation	Float, Dimensionless
Treatment/usage	Define the beam plane to be that of the surface of constant phase[221] at the optical exit aperture. Further, allow that contours of constant power are approximately elliptical. [circular is the degenerate case of elliptical with equal major and minor axes]. Analogously to specifying the detector array geometry by two vectors, the transmit beam is characterized by two vectors. Their cross product determines the beam axis and the major axis determines the orientation of the beam pattern about the beam axis. The direction of the SC.Pulse.BmMinorAxis.TileNum(il,ip,j) is along the far field minor axis direction, orthogonal to SC.Pulse.BmMajorAxis.TileNum(il,ip,j).
Governing quality metrics	
See Also	SC.Pulse.ArrayMajorAxis.*TileNum*(il,ip,j) PD.HiIndx.Laser(1)

20.5.44 SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip, ie, k) [L1,2]

Brief Definition

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) provides information to make lidar responses uniform over the sensing array and collection period. Consists of corrections to be applied to the received range data vector.

indices

The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the receive intensity correction factor data array in the data tag.

TileNum, though part of a tag name, also acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. *TileNum* acts in pulse oriented tiles such as SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) to enable blocking of data into pulse ranges.

Index, ie, ranges from PD.LoIndx.Array.Elem(ia) to PD.HiIndx.Array.Elem(ia), that is, generally from 1 to the number of array elements for array ia. [Certain implementations may choose to base array indexing at 0. The physical association of hardware detector elements to the element index sequence ie must be consistent throughout any single data set.] The correct array index value, ia, to use is that associated with the channel indicated by the value of *ChNum*. This is provided in SC.Channlzn.Descr(j,1), which carries the channel defining array number. The number of elements is also provided directly by SC.Array.LatticeDesc(ia,1). Note that both the channel and the array are used to identify the element sensitivity data. Although there is only one array per channel, a single array could participate in multiple channels with differing values of sensitivity.

Index, ip[222], ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2), that is, from the first pulse (IPP number) in the tile to the last pulse in the tile.

[222] While the index, “ipi”, replaces “ip” at L-2 for wave form tiles so that sparse pulse wave form population is possible, we use the index ip here to indicate that every pulse has potential sensitivity information to be relayed. This may have been used to create point data intensity data. When the sensitivity is not changing with pulse number, the repeated dat strategy would be used to reduce the size of this array.

Index, k, ranges from 1 to 3.

Units/representation Variable, see treatment usage section, Float

Treatment/usage This data can be populated for both GM and LM family lidars. The interpretations are slightly different depending on whether the lidar is operating in Geiger Mode or in the Linear Mode family (includes pulse compression). The k index breakout below explains these differences.

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) provides a factor for each channel element, and for each pulse, that when used to multiply each fast time sample of received range vector data (see LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)) for the corresponding pulse, channel and element, makes the lidar instrument response uniform spatially and in slow time. After application, uncertainty due to instrumentation accuracy effects on the measured intensity is found in SC.WF.Rx.IntnsyUncty.

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) provides

LM: sensitivity <u>at operating temp and bias</u> [current/incident power]

GM: 90th percentile photons/sec to trigger GM,

<u>Relation to photo-detection efficiency:</u>

The photo-detection efficiency (PDE) of a photon detector array represents the effective conversion factor between incident photons and photo-electrons produced by the device. The PDE is a function of the quantum efficiency QE, avalanche probability (P_a) and geometrical efficiency (GE) as follows,

$$PDE = QE(\lambda) \times P_a(V) \times GE \quad (1)$$

QE is dependent on the properties of the solid state detector material and on the wavelength of incident photons.

P_a is dependent on the electric field strength inside the device and thus is dependent on the applied over-bias to the detector elements.

The GE is the ratio of sensitive to insensitive area and thus depends on the design and layout of the pixels only, and is thus fixed for a specific detector array.

PDE is thus a mean statistic and therefore unable to act as an envelope of nearly assured detection events. The inclusion of the geometry factor is tailored to uses of the array for aggregate detection where imaging and calibration of pixels in an image is not a high priority. To provide a metric that is more focused on supporting the ability to process data from real sensors to make them act as more ideal uniform imagers, we not only provide element specific data, but use 90th percentile values in incident photon flux to characterize the events that are likely to have initiated a nonlinear response in the focal plane detector array. So the GE factor is excluded and the QE and P*a* effects are combined. We also express the sensitivity in terms of incident flux or power on an individual detector element's look direction. Thus, any optical bench vignetting is also incorporated into the element level sensitivity data.

As discussed in the definition paragraph for SC.WF.TxProfile.Dat(il,j), there is a convenient option to incorporate the effects of pulse dependent transmit power fluctuations in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1). The

correct laser for the channel indicated by *ChNum* may have fluctuations that affect the relative calibration of return light levels. Note well: the value of the index, ip, in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) corresponds to the IPP in which the returns are recorded, which is not necessarily the same IPP in which the illuminating pulse for those returns was transmitted . When SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) is used to also track transmit power fluctuations, the processor that originally populates SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) {preferably that is the enterprise L-1 writer} must incorporate a pulse aligning shift between the transmitted pulse and the receiving pulse corresponding to the ambiguity number. Once the Tx and Rx effects have been combined, they cannot be unmixed, so conveying Tx fluctuations via SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) should be done only when the certainty that the ambiguity number is correct is quite high or that an error in ambiguity number will not cause violation of the error budget for relative calibration. The metadata indication that SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) is being used to convey the pulse dependent transmit power levels is SC.Laser.IDat(il,5) = 1 and SC.WF.TxProfile.Flags(4) = 0. Other combinations of those values lead to either no Tx power fluctuation tracking or (the currently deferred[223]) independent tracking of pulse power without the ambiguity number shift. If SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) were perfect, the only remaining uncertainty in the scene reflectivity is due to scene and propagation effects. [224]

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,2) provides for
LM and GM a scaling factor that multiplies each entry in the cross talk data array, SC.Array.CrossTalk.*Ia*(j,k,1) to provide element and pulse dependence for the cross talk data.

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,3) provides “dark response”
LM: current when incident photon flux is zero to the detector array
GM: probability of avalanche when incident photon flux is zero to the detector array

[223] We will add SC.Pulse.TxPeakTracker(il,ip) in a later release of the CMMD L1 and L2 documents.

[224] It is understood that the beam pattern, SC.Laser.Beam(il,i,j) will also be used. A complete radiometric error analysis would incorporate orientation errors of the beam pattern. Rigorous Lidar radiometric calibration capability is several technology generations away (as of 2011). When that era is approached, the CMMD may be given another couple of data items to support such a process. We currently defer on the basis of impracticability such items as pulse dependent beam pattern uncertainty, beam to receive plane orientation uncertainties, etc.

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,4) provides "dark response" after a primary excitation. This value differs from SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,3) in that it includes the effect of (laboratory) measured afterpulsing contribution to the detectors' dark response. This supports correcting noise compensations for fast time intervals that are subject to afterpulsing.

LM: after pulsing regime current when incident photon flux is zero to the detector array element
GM: after pulsing regime probability of avalanche when incident photon flux is zero to the detector array element

After-pulsing is caused by electrons from an avalanche response being trapped in intermediate energy levels that are released while or after the pixel is reset. For single quench per pulse GM operation the effect is generally negligible and to the extent that it is present, the impact is carried in the dark current level provided by SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,3). For a mulitple quench GM detector array mode, there may be a heightened dark current level for short times after a primary avalanche event in a given detector which is incorporated in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,4).

If SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) is not populated, processing should proceed as if this information is unknown.

Lidar is an active system. Accounting for its spatial sensitivity variation to scene reflectivity would incorporate the information in SC.Laser.Beam(il,i,j) as well.

SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) can be considered to provide a matrix of data values for each pulse. A row of the matrix for each element has the 3 columns for the 3 sensitivity vector data items. There is a strong likelihood that either only a single value of the sensitivity matrix will be given for a data set that is intended to serve all pulses, or that the sensitivity data matrix will vary slowly enough that many consecutive pulses will have repeated values. To eliminate the redundancy for repeated values of the matrix for multiple pulses, a simplified 3-D data cube reduction scheme is supported by PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg and PD.IntTileMap.Snstvty.Reps.*ChNum.TileNum* (j). The scheme is described in the definition paragraph for PD.IntTileMap.Snstvty.*ChNum.TileNum*.Flg.

Governing quality metrics

See Also	LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) SC.Channlzn.Descr(j,1) PD.IntTileMap.Snstvty.*ChNum*.*TileNum*.Flg SC.Array.CrossTalkQ1.*Ia*(j,k,1) SC.Laser.Beam(il,i,j)

20.5.45 SC.Pulse.FilterCode.*TileNum*(ip, j) [L1,2,3,4,5]

Brief Definition	SC.Pulse.FilterCode.*TileNum*(ip, j) carries pulse dependent filter identification in the case that a filter wheel or analogous mechanism is used to provide spectral diversity from pulse to pulse.
indices	Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles. The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.FilterCode.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.FilterCode.*TileNum*.Flg >0 i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume. *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.
Units/representation	Short IntU, dimensionless
Treatment/usage	Alternating the filter placed in the optical path from pulse to pulse is a design alternative to beam splitting and multiple detector arrays with separate sensitivities or filters. In the case that pulse dependent filtering of the incident photon flux is used, SC.Pulse.FilterCode.*TileNum*(ip, j) takes on the value of a filter code, ifc, defined in the array, SC.Array.FilterDat(ifc,j) appropriate for the pulse filter setting. Note that if two filters are used, one for odd pulses, the other for even pulses, this array could be very large if it were actually in pulse order[225].
Governing quality metrics	
See Also	SC.Array.FilterDat(ifc,j) PD.IntTileMap.FilterCode.*TileNum*.Flg

[225] Sorting the data on the filter number and using repetition factors can lead to drastic data volume reduction. Also, using filters as a dedicated tile parameter would permit skipping a huge data array of toggling values of one parameter. Those are the only means designed into the CMMD to squeeze out this kind of redundancy.

20.5.46 SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j) [L1,2,3,4,5]

Brief Definition

SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j) gives the channel specific effective optical position of the illuminating laser exit aperture in the ECEF frame.

indices

Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2) where j_tile corresponds to *TileNum* via PD.TileMap.TileNum(j_tile). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
The index, j, ranges from 1 to 3 covering ECEF X,Y,Z components respectively.

The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the sensor position data array in the data tag.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. Pulse oriented tiles carry information for a specific interval of slow time within a collection operation.

Units/representation

Meters, Integer variable size (see PD.Wordsize.Sensor Pos)

Treatment/usage

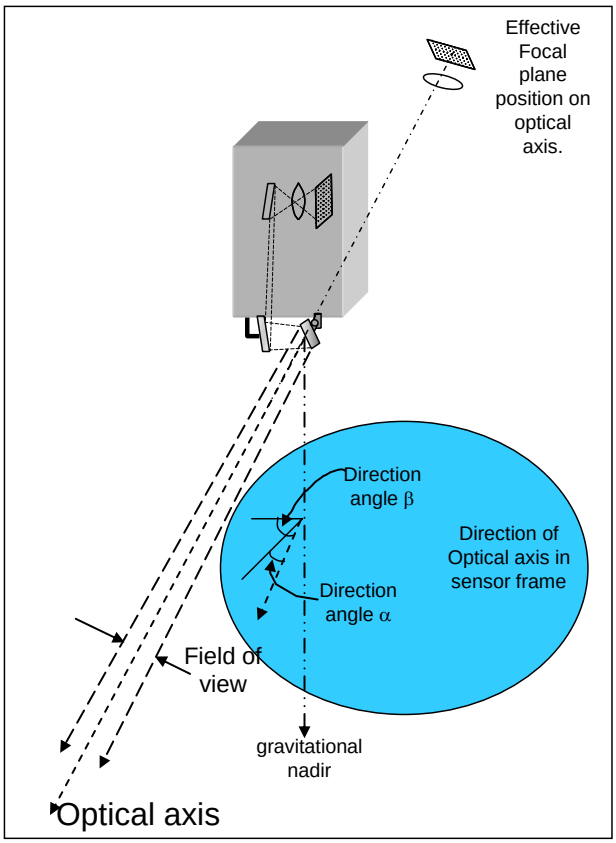
In the following discussion, we use the term “effective optical position”. This treats folded optical benches and moving mirrors and gimbals as if there is a straight line of sight from the center of a (moving) focal plane array to the center of the instantaneously viewed target scene. The effective optical position of the illuminator accounts for optical bench lever arm (including gimbal offsets, mirror positions/angles etc.) and platform motion effects to create a position which lies on the axis of the ray through the center of the illuminating laser associated with *ChNum* and the center of an effective main sensor light exit aperture. The effective optical position of a channel’s illuminator and the effective (transmit) axes’ direction data determine the ray of the illuminating optical axis for each channel.

Use of this definition of illuminator position for each channel places the burden of detailed understanding of the sensor design on the enterprise L-1 writer {or the source of whatever the first CMMD compliant Lidar Data set to

be presented to the PED is}so that a straightforward process can be used to compute lidar point positions which is formally the same for all VRP producing lidars.

Reconstruction of ECEF illuminator channel positions in meters is done in the same way as for the lidar points.
The channel *ChNum* array position in ECEF is found from the linear transformation set:

Xmeters(ip)=Xdat(ip) * X_Fac + X_Offset,
Ymeters(ip)=Ydat(ip) * Y_Fac + Y_Offset,
Zmeters(ip)=Zdat(ip) * Z_Fac + Z_Offset,



Where Xdat(ip)= SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,1)
Ydat(ip)= SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,2)
Zdat(ip)= SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,3)

X_Fac= SC.Swath.Pos.X_Fac
Y_Fac= SC.Swath.Pos.Y_Fac
Z_Fac= SC.Swath.Pos.Z_Fac

The sensor position scale factors are the same for all pulse tiles.

X_Offset=SC.Pulse.TxPos.X_Offset(j_tile)
Y_Offset=SC.Pulse.TxPos.Y_Offset(j_tile)
Z_Offset=SC.Pulse.TxPos.Z_Offset(j_tile)
} The sensor position offsets are separate constants for each tile.

The position, (Xmeters(ip), Ymeters(ip), Zmeters(ip)), computed per the above description is the point, which must be used in conjunction with the round trip time calculation and receiver position and look direction to compute the XYZ of a point return.

SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) would apply to the receiver only in the bistatic case where the illuminator position is provided by SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j). For conventional monostatic lidar, SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j) is not populated.

Governing quality metrics: Percentage error in value

See Also: PD.TileMap.PulseRange(j_tile,j) SC.Pulse.Pos.X_Offset(j_tile) SC.Swath.Pos.X_Fac PD.Wordsize.Sensor Pos

20.5.47 SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) [L1,2,3,4,5]

Brief Definition: SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) provides a sequence of IPP durations in command clock counts that is repeatedly used in a collection operation.

indices: Index, ipn, ranges from PD.LoIndx.IPP(j=*PulseSeq*[226]), to PD.HiIndx.IPP(j=*PulseSeq*). It is the pulse index within a defined pulse duration sequence and ranges from 0 to one less than the number of IPPs in the pulse duration sequence. A single pulse duration sequence is representative of what may be used for a "breathing" PRF over a scan. Everytime a scan is executed with this sequence, the same number of pulses, that is the same range of "ipn" is used. However, the actual IPP number counted from the start of the collection operation (we use "ip" for this) keeps incrementing without starting over for each scan.

PulseSeq, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Dimensionless, UInt32

Treatment/usage: The ipnth element of SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) is the duration of

[226] The notation "j=*PulseSeq*" means that the integer j is the conversion of the numeric character string, *PulseSeq*.

the ipnth IPP in the pulse sequence whose identifier is given by the evaluated character string, *PulseSeq*. The dimensionless units of IPP duration are the number of command clock counts in the IPP. Those durations can be converted to seconds via multiplication by SC.Swath.TimeStep. Examples of two possible such pulse duration sequences are illustrated in the following table.

ipn	Durations *PulseSeq*=1	Durations *PulseSeq*=2
0	1600	24800
1	1602	24800
2	1604	24810
3	1606	24810
4	1608	24820
5	1610	24820
6	1612	24830
7	1614	24845
8	1616	24845
9	1618	24860
10	1620	24860
11		24900
…		
1000		52340

SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) is cited by SC.Pulse.TxCount.*Il.TileNum*(ip,2) when it is populated with negative numbers. The magnitude of such a negative number is the integer conversion of the numeric character string, *PulseSeq*. By matching the tags, SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn), to the codes in SC.Pulse.TxCount.*Il.TileNum*(ip,2), the IPP start time sequence for the entire collection operation can be reconstructed.

Governing quality metrics

See Also PD.HiIndx.IPP(j) SC.Swath.TimeStep

20.5.48 SC.Pulse.PolCode.*TileNum*(ip, j) [L1,2,3,4,5]

Brief Definition SC.Pulse.PolCode.*TileNum*(ip, j) carries pulse dependent polarizer identification in the case that a filter wheel or analogous mechanism is used to provide polarization variation from pulse to pulse.

indices Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges

from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.PolCode.TileNum.Flg set to zero or not populated. When PD.IntTileMap.PolCode.TileNum.Flg >0 i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Short IntU, dimensionless

Treatment/usage: Alternating the polarizer placed in the optical path from pulse to pulse is a design alternative to beam splitting and multiple detector arrays with separate sensitivities or filters. In the case that pulse dependent polarization filtering of the incident photon flux is used, SC.Pulse.PolCode.*TileNum*(ip) takes on the value of a polarizer code, ipc, defined in the array, SC.Array.PolDat(ipc,j) appropriate for the pulse polarizer setting.
Note that if two filters are used, one for odd pulses, the other for even pulses, this array could be very large if it were actually in pulse order.

Governing quality metrics:

See Also: SC.Pulse.FilterCode.*TileNum*(ip,j) PD.IntTileMap.PolCode.TileNum.Flg

20.5.49 SC.Pulse.Pos.X_Offset(j_tile) [L1,2,3,4,5]

Brief Definition: SC.Pulse.Pos.X_Offset(j_tile) carries the x- intercept term in a linear transformation to be applied to the lidar sensor position data stored in a tile to form actual position data in the ECEF frame.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges over the pulse oriented tiles from PD.LoIndx.Pulse to PD.HiIndx.Pulse.

Units/representation: Meters, Double Float

Treatment/usage: The x-component of an ECEF lidar sensor position, X_{meters}, is given by a linear transformation of the form:
X_{meters}=Xdat * X_ScaleFac +X_Offset,
where X_Offset is given by SC.Pulse.Pos.X_Offset(j_tile) for every position in the tile indexed by j_tile. The lidar pulse indexed position ("Xdat" above) stored in the product for input to the above transform is supplied via

SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,1) and the scaling factor,("X_ScaleFac" above) is found in SC.Swath.Pos.X_Fac.
The intent of the offsets stored in SC.Pulse.Pos.X_Offset(j_tile) is to permit reduction of the wordsize for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j). The precision of SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) need only be enough to span the position range[227] of the pulse oriented tile number, *TileNum*.

SC.Pulse.Pos.X_Offset(j_tile) would apply only to the receiver in a bistatic collection, which would be when the illuminator position offset is given by SC.Pulse.TxPos.X_Offset(j_tile).

Governing quality metrics

See Also: SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Swath.Pos.X_Fac PD.TileMap.TileNum(j_tile) PD.HiIndx.Pulse

20.5.50 SC.Pulse.Pos.Y_Offset(j_tile) [L1,2,3,4,5]

Brief Definition: SC.Pulse.Pos.Y_Offset(j_tile) carries the y- intercept term in a linear transformation to be applied to the lidar sensor position data stored in a tile to form actual position data in the ECEF frame.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges over the pulse oriented tiles from PD.LoIndx.Pulse to PD.HiIndx.Pulse.

Units/representation: Meters, Double Float

Treatment/usage: The y-component of an ECEF lidar sensor position, Y_{meters}, is given by a linear transformation of the form:
Y_{meters}=Ydat * Y_ScaleFac +Y_Offset,
where Y_Offset is given by SC.Pulse.Pos.Y_Offset(j_tile) for every position in the tile indexed by j_tile. The lidar pulse indexed position ("Ydat" above) stored in the product for input to the above transform is supplied via SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,2) and the scaling factor,("Y_ScaleFac" above) is found in SC.Swath.Pos.Y_Fac.
The intent of the offsets stored in SC.Pulse.Pos.Y_Offset(j_tile) is to permit reduction of the wordsize for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j). The precision of SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) need only be enough to span the position range of the pulse oriented tile number, *TileNum*.

SC.Pulse.Pos.Y_Offset(j_tile) would apply only to the receiver in a bistatic

[227] This implies that the writing algorithm is correctly flowing down the accuracy requirements and that the positional increment implied by SC.Swath.Pos.X_Fac is matched to the precision of SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) given in PD.WordSize.SensorPos and the range of position for the sensor during the tile duration. The duration of the tile in pulses can be derived from PD.TileMap.PulseRange(j_tile,k).

collection, which would be when the illuminator position offset is given by SC.Pulse.TxPos.Y_Offset(j_tile).

Governing quality metrics

See Also SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Swath.Pos.Y_Fac PD.TileMap.TileNum(j_tile) PD.HiIndx.Pulse

20.5.51 SC.Pulse.Pos.Z_Offset(j_tile) [L1,2,3,4,5]

Brief Definition SC.Pulse.Pos.Z_Offset(j_tile) carries the z- intercept term in a linear transformation to be applied to the lidar sensor position data stored in a tile to form actual position data in the ECEF frame.

indices The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges over the pulse oriented tiles from PD.LoIndx.Pulse to PD.HiIndx.Pulse.

Units/representation Meters, Double Float

Treatment/usage The z-component of an ECEF lidar sensor position, Z_{meters}, is given by a linear transformation of the form:
Z_{meters}=Zdat * Z_ScaleFac +Z_Offset,
where Z_Offset is given by SC.Pulse.Pos.Z_Offset(j_tile) for every position in the tile indexed by j_tile. The lidar pulse indexed position ("Zdat" above) stored in the product for input to the above transform is supplied via SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,3) and the scaling factor,("Z_ScaleFac" above) is found in SC.Swath.Pos.Z_Fac.
The intent of the offsets stored in SC.Pulse.Pos.Z_Offset(j_tile) is to permit reduction of the wordsize for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j). The precision of SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) need only be enough to span the position range[227] of the pulse oriented tile number, *TileNum*.

SC.Pulse.Pos.Z_Offset(j_tile) would apply only to the receiver in a bistatic collection, which would be when the illuminator position offset is given by SC.Pulse.TxPos.Z_Offset(j_tile).

Governing quality metrics

See Also SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Swath.Pos.Y_Fac PD.TileMap.TileNum(j_tile) PD.HiIndx.Pulse

20.5.52 SC.Pulse.RecordStrt.*TileNum*(ip, j) [L1,2]

Brief Definition	SC.Pulse.RecordStrt.*TileNum*(ip, j) is the number of (command) clock counts after[228] the start of an IPP when the command is issued to commence recording output of the receive electronics. Also called gate delay.
indices	Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles. The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.RecordStrt.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.RecordStrt.*TileNum*.Flg >0 i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume. *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.
Units/representation	Dimensionless, UInt32
Treatment/usage	Receiver commands can only be issued on integral clock counts. SC.Pulse.RecordStrt.*TileNum*(ip, j) is the commanded count within an IPP when the order is issued to open the receive window(s)[229] for data capture and recording. Pulse dependence allows variation with scan phase so that gates can track expected range to scene content. The time delay to the actual recording is not necessarily an integral number of command clock counts and the finer granularity time tracking of record start is accomplished with the addition of SC.Swath.RecordDelay to SC.Pulse.RecordStrt.*TileNum*(ip, j). The precise time relative to the start of a collection operation that the received

[228] If we say the clock count at the start of an IPP is defined to be count 0, then the clock count that triggers the issue of the 'start recording' command is in SC.Pulse.RecordStrt.*TileNum*(ip, j).

[229] This might also be termed the range gate start data on a pulse basis. A system with automatic range gate adjustment might be generating a large number of record start times, but usually in runs of many pulses for each. The "(s)" is to indicate that there may be multiple windows being opened. One could argue that, even if simultaneously opened, each channel has its own receive window. Also, a multi-quench Geiger mode system may use this data item to indicate when the recording process begins for an IPP, but there will be discrete intervals when the sensor is actually able to receive photons it can record. Finally, the "ip" index indicates that SC.Pulse.RecordStrt.*TileNum*(ip, j) is a list of record start times for all of the IPPs in a tile. Each has its own window and the delay to opeing that window may be unique to each pulse.

returns begin to be recorded in the ipth IPP is given by

Time of Rx $start_{ip}$=$TxTime_{ip}$ +$record_start_{ip}$ + record_delay

where:

- $TxTime_{ip}$ is obtained by summing over the preceding IPP interval durations.
- $record_start_{ip}$ is from SC.Pulse.RecordStrt.*TileNum*(ip, j)
- record_delay is from SC.Swath.RecordDelay

This time may be desired to synch up with external independent data sources. Typically of interest is the receive window start time time relative to the start of the IPP, ($record_start_{ip}$ + record_delay).

Governing quality metrics

See Also SC.Swath.RecordDelay PD.IntTileMap.RecordStrt.*TileNum*.Flg LD.WFClp.Tstart.*TileNum*(ipi,iray)

20.5.53 SC.Pulse.Scan.Num.*TileNum*(ip, j) [L1,2]

Brief Definition SC.Pulse.Scan.Num.*TileNum*(ip, j) maps a pulse into a scan number.

indices Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.

The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.Scan.Num.*TileNum*.Flg set to zero or not populated. When IntTileMap.Scan.Num.*TileNum*.Flg >0, i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation Dimensionless, UInt32

Treatment/usage Scans and scan parameterization are discussed in 10.1.4.2 and 10.1.2.4. The value of SC.Pulse.Scan.Num.*TileNum*(ip,j) for a given ip is the scan index, "is"[230] that the pulse belongs to. Thus, SC.Pulse.Scan.Num.*TileNum*(ip, j) is

[230] Per 10.1.4.2, a single lidar has a single assignment of pulse number to scan number. Spatial channelization could make "is" alone ambiguous with respect to scan ground coverage. In that case , the coverage of the scan parameterization could be channel dependent. However, all the channels have the same scan to pulse (and pulse to scan) assignment.

a look-up table. Many consecutive "ip" values will have the same "is" value so the tiles for SC.Pulse.Scan.Num.*TileNum*(ip, j) are good candidates for the internal tile mapping procedure of 10.2.1.5 to reduce the array size. The table conveyed in SC.Pulse.Scan.Num.*TileNum*(ip,j) can be used to determine the scan index for a given pulse number, or it can be used to determine a range of pulses for a given scan index. Per 10.1.4.2, the scan index increments consecutively throughout the data set and thus the first scan index of tile N+1 of SC.Pulse.Scan.Num.*TileNum*(ip,j) is either the same as the last scan index of tile N or is 1 greater, indicating whether the pulses in the tiles were divided on scan boundaries (increments) or not (stays the same).

Governing quality metrics: Correct values populated.

See Also: PD.IntTileMap.Scan.Num.*TileNum*.Flg

20.5.54 SC.Pulse.Scan.Phase.*TileNum*(ip, j) [L1,2]

Brief Definition indices: SC.Pulse.Scan.Phase.*TileNum*(ip, j) gives the phase of the scan by pulse. Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.Scan.Phase.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.Scan.Phase.*TileNum*.Flg >0, then there are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume. It would be a rare situation for the data sort to favor repetition factors for the scan phase data.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Float, radians

Treatment/usage: SC.Pulse.Scan.Phase.*TileNum*(ip, j), the scan phase on a pulse basis, must be the same for all channels to assure correlation of geolocation errors across channels. When channel errors are not correlated, the system should be treated as multiple lidars rather than a multichannel lidar.
Per the discussion of 10.1.2.4, steering angles relative to the sensor frame are

varied periodically to positions on an elliptical scan cone model by selection of a value for the scan phase. The scan cone, which could have a minor axis of zero to collapse the scan pattern to be confined to a plane for a single period, is used to determine where a pulse is directed relative to other pulses in neighboring time intervals. It is important to understand that the precise directions of the transmit and receive optical axes are to be computed on a pulse basis from vectors defining the effective detector and beam planes and those calculations are directly supported in the data without having to evaluate the scan parameters for a pulse. [see for example, SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j), SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j)] The scan cone parameters, evaluated with the input of the pulse specific scan phase values, describe the *plan* for the lidar beam control relative to the sensor position, but may not incorporate special corrections to the actual optical axis orientation in the earth frame provided in the pulse dependent array axis data. So, one should not use the scan parameters and pulse index values to perform precise optical axis direction calculations. SC.Pulse.Scan.Phase.*TileNum*(ip, j) is the basis for constructing the scan cone from its parameters and evaluating the instantaneous line of sight of the receiver on that cone. Intersecting the scan cone with a simple model of the ground/scene traces the scans and can be valuable in understanding the collection patterns within a lidar product. Recognizing key scan phase value transitions can help determine if a pulse is at the ends of a scan and what the rate of overlap change of adjacent pulse frames is. It is likely that the sensor may command beam orientations via a scan phase and cone parameters, but unless the ArrayMajorAxis.*TileNum*(ip, j) and ArrayMinorAxis.*TileNum*(ip, j) data have been populated by an enterprise level-1 writer by evaluation of scan parameters, these data will not be precisely equivalent. It is expected that the actual beam control that drives the ArrayMajorAxis.*TileNum*(ip, j) data includes corrections for platform orientation variations that will be added to the nominal periodic scan sweep articulations.

Governing quality metrics

Valid values are 0 to 2π[231].

[231] Technically 6.283184. The single precision value of 2pi is 6.283185, but that would wrap to 0, so the next smaller value representable as a single float is the maximum value for the phase.
A phase value of -10, clearly outside the 2π cycle is recommended for uses in other data items where a N/A for scan phase is wanted. An example is a value of phase in a dedicated tile table, e.g. PD.TileMap.DedictdTile.Cnstrnt1.Val(j), where a particular tile group may not have an effective scan phase value as a constraint, but others do. However, only valid values for scan phase should ever appear in SC.Pulse.Scan.Phase.*TileNum*(ip, j).

See Also
SC.Pulse.ArrayMajorAxis.*TileNum*(ia,ip,j)
SC.Pulse.BmMajorAxis.*TileNum*(il,ip,j)
PD.IntTileMap.Scan.Phase.*TileNum*.Flg

20.5.55 SC.Pulse.SensorAxisX.*TileNum*(ip,j) [L1,2]

Brief Definition
SC.Pulse.SensorAxisX.*TileNum*(ip,j) gives the ECEF components of the sensor frame X axis from which beam steering angles are measured.

indices
Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.
The index, j, ranges from 1 to 3 covering ECEF X,Y,Z components respectively.

Units/representation
Dimensionless, Float

Treatment/usage
The sensor frame is attached to the body of the sensor and moves as the platform and the sensor mount respond to external forces. This is distinct from the effective frame of the focal plane detector array, and provides a reference from which the instantaneous look direction (optical) axis of the sensor is to be measured. The orientation of the sensor frame needs to be provided for each pulse, thus the "ip" index is used. The direction of the sensor frame x-axis is chosen such that the sensor x-axis cross product with the sensor y-axis, $\mathbf{X} \times \mathbf{Y}$ points into the look hemisphere of the sensor.

To be clear, SC.Pulse.SensorAxisX.*TileNum*(ip,j) is part of the scan cone based beam steering model and is not directly used in computation of ray based lidar point positions. SC.Pulse.SensorAxisX.*TileNum*(ip,j) is part of establishing the frame in which the scan cone parameters are defined at an instant in slow time. Without that, projecting the scan cone to the ground surface would not be possible.

Governing quality metrics

See Also
SC.Pulse.SensorAxisY.*TileNum*(ip,j)

20.5.56 SC.Pulse.SensorAxisY.*TileNum*(ip,j) [L1,2]

Brief Definition	SC.Pulse.SensorAxisY.*TileNum*(ip,j) gives the ECEF components of the sensor frame Y axis from which beam steering angles are measured.
indices	Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles. *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. The index, j, ranges from 1 to 3 covering ECEF X,Y,Z components respectively.
Units/representation	Dimensionless, Float
Treatment/usage	The sensor frame is attached to the body of the sensor and moves as the platform and the sensor mount respond to external forces. This is distinct from the effective frame of the focal plane detector array, and provides a reference from which the instantaneous look direction (optical) axis of the sensor is to be measured. The orientation of the sensor frame needs to be provided for each pulse, thus the "ip" index is used. The direction of the sensor frame y-axis is chosen such that the sensor x-axis cross product with the sensor y-axis, **X** × **Y** points into the look hemisphere of the sensor. To be clear, SC.Pulse.SensorAxisY.*TileNum*(ip,j) is part of the beam steering model and is not directly used in computation of ray based lidar point positions. SC.Pulse.SensorAxisY.*TileNum*(ip,j) is part of establishing the frame in which the scan cone parameters are defined at an instant in slow time. Without that, projecting the scan cone to the ground surface would not be possible.
Governing quality metrics	Percentage error in value
See Also	SC.Pulse.SensorAxisX.*TileNum*(ip,j)

20.5.57 SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) [L1,2,3,4,5]

Brief Definition	SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) gives the channel specific effective optical position of the sensing array in the ECEF frame.
indices	Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to

PD.TileMap.PulseRange(j_tile,2) where j_tile corresponds to *TileNum* via PD.TileMap.TileNum(j_tile). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.
The index, j, ranges from 1 to 3 covering ECEF X,Y,Z components respectively.

The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the sensor position data array in the data tag.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. Pulse oriented tiles carry information for a specific interval of slow time within a collection operation.

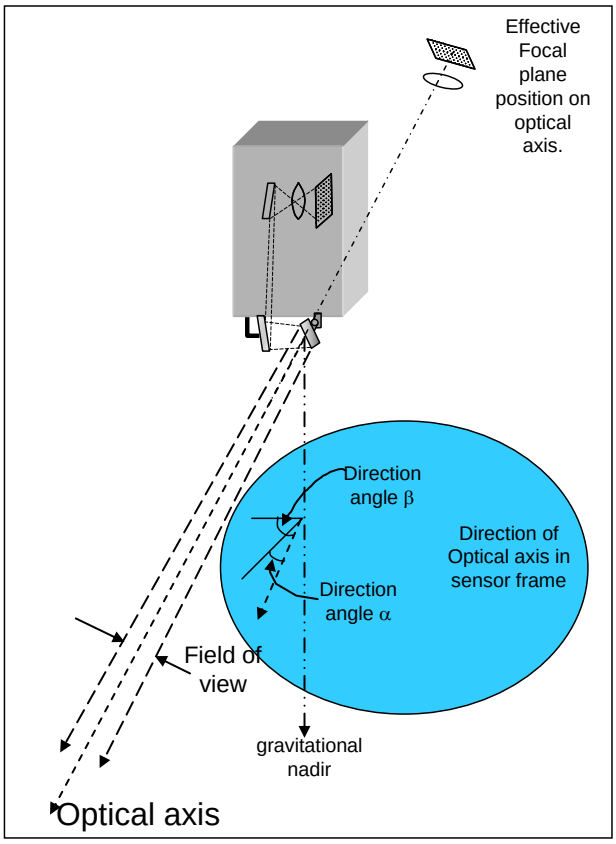
Units/representation
Meters, Integer variable size (see PD.Wordsize.Sensor Pos)

Treatment/usage
In the following discussion, we use the term “effective optical position”. This treats folded optical benches and moving mirrors and gimbals as if there is a straight line of sight from the center of a (moving) focal plane array to the center of the instantaneously viewed target scene. The effective optical position of the sensor accounts for optical bench lever arm (including gimbal offsets, mirror positions/angles etc.) and platform motion effects to create a position which lies on the axis of the ray through the center of the focal plane detection array associated with *ChNum* and the center of an effective main sensor light entrance aperture. The effective optical position of a channel’s array center and the effective (receive) array axes’ direction data determine the ray of the optical axis for each channel.

Use of this definition of sensor position for each channel places the burden of detailed understanding of the sensor design on the enterprise L-1 writer {or the source of whatever the first CMMD compliant lidar data set to be presented to the PED is}so that a straightforward process can be used to compute lidar point positions which is formally the same for all VRP producing lidars.

Reconstruction of ECEF sensor channel positions in meters is done in the same way as for the lidar points.
The channel *ChNum* array position in ECEF is found from the linear transformation set:

Xmeters(ip)=Xdat(ip) * X_Fac + X_Offset,
Ymeters(ip)=Ydat(ip) * Y_Fac + Y_Offset,
Zmeters(ip)=Zdat(ip) * Z_Fac + Z_Offset,



Where Xdat(ip)= SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,1)
Ydat(ip)= SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,2)
Zdat(ip)= SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,3)

X_Fac= SC.Swath.Pos.X_Fac
Y_Fac= SC.Swath.Pos.Y_Fac
Z_Fac= SC.Swath.Pos.Z_Fac
} The sensor position scale factors are the same for all pulse tiles.

X_Offset=SC.Pulse.Pos.X_Offset(j_tile)
Y_Offset=SC.Pulse.Pos.Y_Offset(j_tile)
Z_Offset=SC.Pulse.Pos.Z_Offset(j_tile)
} The sensor position offsets are separate constants for each tile.

The position, (Xmeters(ip), Ymeters(ip), Zmeters(ip)), computed per the above description is the point, which when added to the sensor based return point position derived from transformation of (angle, angle, range) to XYZ, yields an ECEF XYZ position for a lidar point.

SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) would apply to the receiver only in the bistatic case where the illuminator position is provided by SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j). For conventional monostatic lidar, SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) is the position of the full sensor.

Governing quality metrics: Percentage error in value

See Also: PD.TileMap.PulseRange(j_tile,j) SC.Pulse.Pos.X_Offset(j_tile) SC.Swath.Pos.X_Fac PD.Wordsize.Sensor Pos

20.5.58 SC.Pulse.TxCount.*Il*.*TileNum*(ip,j) [L1,2,3,4,5]

Brief Definition: SC.Pulse.TxCount.*Il*.*TileNum*(ip,j) is a code supporting the documentation of the duration of all of the IPPs in the collection data underlying the current data set.

indices: "*Il*" while part of the name of the data item, acts as an index. The value of *Il* is the alpha-numeric translation of the illuminator index and ranges from the BCS translation of PD.LoIndex.Laser(1) to PD.HiIndx.Laser(1), that is, over the laser illuminators in the lidar used for this collection. This provides channel dependence for the PRF.

Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset of the global IPP counts is for pulse oriented tiles. The value of j_tile referenced to establish the range of "ip" here must match the value of *TileNum* embedded in the tag, SC.Pulse.TxCount.*Il*.*TileNum*(ip,j).

The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.TxCount.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.TxCount.*TileNum*.Flg >0, i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. The SC.Pulse.TxCount.*Il*.*TileNum*(ip,1) in this case carries the number of times the value of the code in SC.Pulse.TxCount.*Il*.*TileNum*(ip,2) is repeated consecutively. This index supports repetition factor use to reduce data volume.

TileNum, though part of a tag name, acts somewhat as an index. It indicates

which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index. Being that SC.Pulse.TxCount.Il.*TileNum*(ip,j) is a pulse oriented set of tiles addressing the length of IPP intervals throughout the collection operation, the values of *TileNum* are limited to those corresponding to the range PD.LoIndx.Pulse to PD.HiIndx.Pulse.

Units/representation: Dimensionless, Int32

Treatment/usage: Background:

Many lidars are designed as constant PRF instruments.
But, as is explained in the timing model discussions of 10.1.1, this can be a performance limiting design when large areas or multiple nadir angle collections are an important objective. If, for example the range to the scene varies significantly during a collection or, for that matter within a scan, a constraint of constant ambiguity number leads to a breathing PRF. Thus, to provide general support to lidar functionality, the PRF is not treated as the primary metadata parameter for pulse timing; the IPP duration, possibly varying from pulse to pulse is. We seek a simple parameterization of the slow time model so that we can handle an arbitrary pulse timing sequence, a set of repeated pulse timing sequences (relevant when there are scans from more than one "nadir class" that each have a repeated fixed sequence of varying length IPPs, and the simplest special case: constant PRF, which means a single IPP duration for the entire collection operation.

Encoding of Slow time pulse sequences:

In order to convey this information as compactly as possible, SC.Pulse.TxCount.*Il.TileNum*(ip,j) is defined as an integer number that is a code for the IPP transmit interval information.

When SC.Pulse.TxCount.*Il.TileNum*(ip,j) is positive, the value can be directly interpreted as the number of command clock counts within the IPP whose pulse count from the start of the collection operation is given by the value of the index, "ip". Note that the indexing convention here (see indices, above) is for the index counting to advance consecutively through the tiles, with the starting and ending index values for a given *TileNum* provided by PD.TileMap.PulseRange(j_tile,k).

When SC.Pulse.TxCount.*Il.TileNum*(ip,j) is negative, the magnitude of the value is to be interpreted as a pulse sequence number. A pulse sequence number is an identifier for a repeatable sequence of varying length IPPs whose durations are provided in

SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn). The integer valued pulse sequence number from SC.Pulse.TxCount.*Il.TileNum*(ip,j) will match up with the integer translation of a *PulseSeq* (numeric character) substring in the name of a SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) tag. A practical application is when the PRF varies with the relative geometry during a scan. That scan's IPP duration sequence may be repeated 1000 times in succession as the lidar moves over the target scene region.

SC.Pulse.TxCount.*Il.TileNum*(ip,j) can use the internal mapping of consecutive repeated values for the common case of constant PRF. In that case, the length of IPP intervals in clock counts is the positive valued code in SC.Pulse.TxCount.*Il.TileNum*(ip,2) and the number of consecutive pulses with this duration is given in SC.Pulse.TxCount.*Il.TileNum*(ip,1). When we use the repeat factor for reduction of data redundancy,
" ip" indexes the ordinal value of an IPP duration. For constant PRF cases, there is only one IPP duration value and this will be indicated by PD.IntTileMap.TxCount.*TileNum*.Flg =1. Thus index, ip, has a range of only 1 row of data. Since value of the code in this case is positive, SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) need neither be populated nor consulted.

Alternately, if needed, a long list of different pulse time separations can be populated into SC.Pulse.TxCount.*Il.TileNum*(ip,j) to permit computing the start time of any pulse in a series of pulses of varying IPP. This would mean that neither PD.IntTileMap.TxCount.*TileNum*.Flg nor SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) would be populated. A synthetic aperture collection is an example of when this might be applied.

If the PRF breathes during repeated scans, the use case would follow these lines:

- PD.IntTileMap.TxCount.*TileNum*.Flg is set to a positive number. This gives the number of rows (values of index "ip") in SC.Pulse.TxCount.*Il.TileNum*(ip,j). For each consecutive run of a distinct pulse sequence, there will be a row in SC.Pulse.TxCount.*Il.TileNum*(ip,j). For the case of all of the scans having the same IPP duration sequence, there would be only one row.
- Each SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn) will be populated with a data vector. Each element is the duration in command clocks for the ipnth IPP in the repeated sequence. If there were two distinct sequences of IPP durations (as might go with along track scans that are at two different cross track ranges), then we could populate

SC.Pulse.IPP_DurSeq.1(ipn) and SC.Pulse.IPP_DurSeq.2(ipn) with individual IPP duration sequences.

- SC.Pulse.TxCount.*Il.TileNum*(1,1) will tell us the number of times the first sequence is executed consecutively.
- SC.Pulse.TxCount.*Il.TileNum*(1,2) provides the identity of the first sequence. If it is set to -1, then the IPP durations will be found in SC.Pulse.IPP_DurSeq.1(ipn)
- SC.Pulse.TxCount.*Il.TileNum*(2,1) will tell us the number of times the second sequence is executed consecutively.
- SC.Pulse.TxCount.*Il.TileNum*(2,2) provides the identity of the second sequence. If it is set to -2, then the IPP durations will be found in SC.Pulse.IPP_DurSeq.2(ipn)
- SC.Pulse.TxCount.*Il.TileNum*(3,1) will tell us the number of times the third sequence is executed consecutively.
- SC.Pulse.TxCount.*Il.TileNum*(3,2) provides the identity of the third sequence. If it is set to -1, then the IPP durations will be found in SC.Pulse.IPP_DurSeq.1(ipn)
- Etc. depending on the value of PD.IntTileMap.TxCount.*TileNum*.Flg

The time in command clock counts since the start of IPPs (indexed by ip=0 at MD.Collect.StartTime) to the start of the nth IPP is the sum of the SC.Pulse.TxCount.*Il.TileNum*(ip,j) values over all the preceding IPPs. This time can be converted to seconds by multiplying the above sum by the command clock step interval, SC.Swath.TimeStep.
PD.TileMap.T_Range(j_tile,k) gives the temporal range of each tile of data in the data set.
PD.TileMap.PulseRange(j_tile,1) gives the pulse number for the earliest data in the j_tile^{th} tile

Governing quality metrics: Correct value for each value of the index, ip, is populated

See Also: SC.Swath.TimeStep PD.TileMap.T_Range(j_tile,k) PD.TileMap.PulseRange(j_tile,1) PD.IntTileMap.TxCount.*TileNum*.Flg PD.TileMap.StartPulseNum.*TileNum*(j_parent,j,k) SC.Pulse.IPP_DurSeq.*PulseSeq*(ipn)

20.5.59 SC.Pulse.TxDelay.*TileNum*(ip,il) [L1,2]

Brief Definition: SC.Pulse.TxDelay.*TileNum*(ip,il) gives the interval after the command to fire a pulse that the pulse emerges as if from the range of the sensor focal plane.

indices: Index, ip, ranges from PD.TileMap.PulseRange(j_tile,1) to

PD.TileMap.PulseRange(j_tile,2). The pulse index for a collection ranges from 0 to one less than the number of pulses transmitted. Any given tile will have a subset of the total pulses. PD.TileMap.PulseRange(j_tile,k) documents what that subset is for pulse oriented tiles.

Index, il, ranges from PD.LoIndex.Laser(1) to PD.HiIndx.Laser(1), that is over the laser illuminators in the lidar used for this collection.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation
Dimensionless fast time sampling clock counts, UInt16

Treatment/usage
The pulse transmission sequence for a lidar typically begins with the execution of a pulse transmit command that is clock triggered. After a short interval of time, the Laser actually fires.The pulse travels through the lidar optical chain and out the exit aperture. A number of fast data capture clock cycles pass between the command to fire and when the pulse begins to emerge from the same range from the exit aperture as the sensor array lies. Computation of round trip time must be from when the pulse seems to leave the position of the focal plane array to when the pulse is received back at the focal plane[232] [and must include the impact of a possibly non-zero ambiguity number]. Precise knowledge of the time of emergence of the transmitted pulse is given by adding the pulse (and laser) dependent delay SC.Pulse.TxDelay.*TileNum*(ip,il) to the pre-programmed transmit command execution time that can be obtained from SC.Pulse.TxCount.*Il.TileNum*(ip,j).

[232] This will incorporate any delay due to in-sensor extra path length, such as may result from optical bench components that cause the light path to deviate from the ray through the effective sensor position in the effective look direction. Since the receive side delay in SC.Swath.RecordDelay is a constant, time dependent receive path variations that vary on a scale longer than the ambiguity number times the PRF can be included in this term so that roundtrip time calculations predict the range from the mechanical effective sensor position correctly. Higher frequency receive delay variations will be negligible in a properly functioning Lidar.

Round Trip calculation

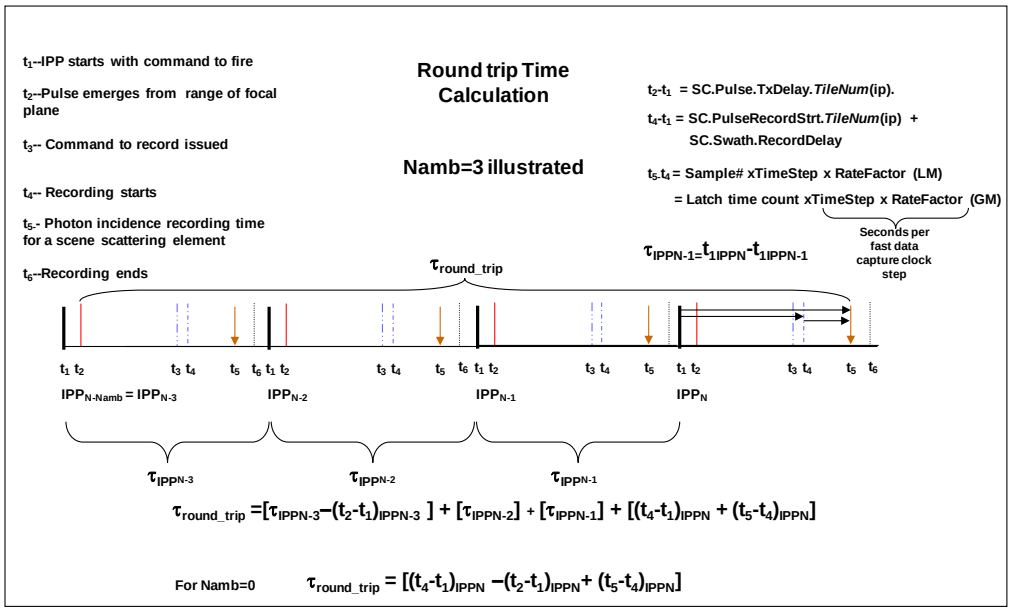


Figure: Fast time markers within several consecutive IPPs in support of ranging calculation. Case of ambiguity number =3 illustrated. Note that computation of t_5 to required accuracy could demand adding in SC.Swath.FineTimeCal.*Ia*(ie).

Governing quality metrics

See Also SC.Pulse.TxCount.*Il.TileNum*(ip,j)

20.5.60 SC.Pulse.TxPos.X_Offset(j_tile) [L1,2,3,4,5]

Brief Definition SC.Pulse.TxPos.X_Offset(j_tile) carries the x- intercept term in a linear transformation to be applied to the lidar illuminator position data stored in a tile to form actual position data in the ECEF frame. Used for bistatic collection only.

indices The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges over the pulse oriented tiles from PD.LoIndx.Pulse to PD.HiIndx.Pulse.

Units/representation Meters, Double Float

Treatment/usage The x-component of an ECEF lidar illuminator position, X_{meters}, is given by a linear transformation of the form:
X_{meters}=Xdat * X_ScaleFac +X_Offset,
where X_Offset is given by SC.Pulse.TxPos.X_Offset(j_tile) for every position in the tile indexed by j_tile. The lidar illuminator pulse indexed position ("Xdat" above) stored in the product for input to the above transform is supplied via SC.Pulse.IllumntrPos.*ChNum.TileNum*(ip,1) and the scaling factor,("X_ScaleFac" above) is found in SC.Swath.Pos.X_Fac.
The intent of the offsets stored in SC.Pulse.TxPos.X_Offset(j_tile) is to permit

reduction of the wordsize for SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j). The precision of SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j) need only be enough to span the position range[233] of the pulse oriented tile number, *TileNum*.

Governing quality metrics

See Also: SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Swath.Pos.X_Fac PD.TileMap.TileNum(j_tile) PD.HiIndx.Pulse

20.5.61 SC.Pulse.TxPos.Y_Offset(j_tile) [L1,2,3,4,5]

Brief Definition: SC.Pulse.TxPos.Y_Offset(j_tile) carries the y- intercept term in a linear transformation to be applied to the lidar illuminator position data stored in a tile to form actual position data in the ECEF frame. Used for bistatic collection only.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges over the pulse oriented tiles from PD.LoIndx.Pulse to PD.HiIndx.Pulse.

Units/representation: Meters, Double Float

Treatment/usage: The y-component of an ECEF lidar illuminator position, Y_{meters}, is given by a linear transformation of the form:
Y_{meters}=Ydat * Y_ScaleFac +Y_Offset,
where Y_Offset is given by SC.Pulse.TxPos.Y_Offset(j_tile) for every position in the tile indexed by j_tile. The lidar illuminator pulse indexed position ("Ydat" above) stored in the product for input to the above transform is supplied via SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,1) and the scaling factor,("Y_ScaleFac" above) is found in SC.Swath.Pos.Y_Fac.
The intent of the offsets stored in SC.Pulse.TxPos.Y_Offset(j_tile) is to permit reduction of the wordsize for SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j). The precision of SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j) need only be enough to span the position range of the pulse oriented tile number, *TileNum*.

Governing quality metrics

See Also: SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Swath.Pos.Y_Fac PD.TileMap.TileNum(j_tile) PD.HiIndx.Pulse

20.5.62 SC.Pulse.TxPos.Z_Offset(j_tile) [L1,2,3,4,5]

Brief Definition: SC.Pulse.TxPos.Z_Offset(j_tile) carries the z- intercept term in a linear transformation to be applied to the lidar illuminator position data stored in a tile to form actual position data in the ECEF frame. Used for bistatic collection only.

indices: The index, j_tile, corresponds to a unique evaluation of *TileNum* as provided by PD.TileMap.TileNum(j_tile) and ranges over the pulse oriented tiles from

[233] This implies that the writing algorithm is correctly flowing down the accuracy requirements and that the positional increment implied by SC.Swath.Pos.X_Fac is matched to the precision of SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) given in PD.WordSize.SensorPos and the range of position for the sensor during the tile duration. The duration of the tile in pulses can be derived from PD.TileMap.PulseRange(j_tile,k) .

	PD.LoIndx.Pulse to PD.HiIndx.Pulse.
Units/representation	Meters, Double Float
Treatment/usage	The y-component of an ECEF lidar illuminator position, Z_{meters}, is given by a linear transformation of the form: Z_{meters}=Zdat * Z_ScaleFac +Z_Offset, where Z_Offset is given by SC.Pulse.TxPos.Z_Offset(j_tile) for every position in the tile indexed by j_tile. The lidar illuminator pulse indexed position ("Zdat" above) stored in the product for input to the above transform is supplied via SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,1) and the scaling factor,("Z_ScaleFac" above) is found in SC.Swath.Pos.Z_Fac. The intent of the offsets stored in SC.Pulse.TxPos.Z_Offset(j_tile) is to permit reduction of the wordsize for SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j). The precision of SC.Pulse.IllumntrPos.*ChNum*.*TileNum*(ip,j) need only be enough to span the position range of the pulse oriented tile number, *TileNum*.
Governing quality metrics	
See Also	SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Swath.Pos.Y_Fac PD.TileMap.TileNum(j_tile) PD.HiIndx.Pulse

SC.Quench.*

SC.Quench.* parameters describe sensor configuration items related to the quenching of Geiger Mode APDs.

20.5.63 SC.Quench.Times(j) [L1,2,3,4,5]

Brief Definition	SC.Quench.Times(j) gives the command clock counts when Quenches of Geiger mode APD arrays occur.
indices	Index j ranges from PD.LoIndx.Quench to PD HiIndx.Quench., that is generally from 1 to the number of quench times used per pulse.
Units/representation	Dimensionless, UInt32
Treatment/usage	For Geiger mode lidars, the times at which the APD focal plane array are quenched are important for understanding the collected data. Simple first generation designs will tend to quench once per pulse before each exposure. Due to the fact that the sensor elements are effectively blind once triggered, multiple quenches can support scenes with scatterers at multiple ranges on any ray, including overcoming some of the impact of range ambiguities, which become more of a challenge as PRFs are increased in order to improve the statistics of the collection. Thus, SC.Quench.Times(j) provides a list of multiple quench times, which can be as short as a single time for current single quench sensors.

	The quench time for the jth entry in this array is the number of command clock counts after the start of the IPP (command to fire issued)[234] that the quench command is issued for the jth quench. Thus, if the IPP is assumed to start on command clock value N, and the first quench command is issued on clock N+20, the second quench command is issued on clock N+200, then SC.Quench.Times(1) would be set to 20 and SC.Quench.Times(2) would be set to 200.
Governing quality metrics	Correct values populated
See Also	SC.Quench.Outage PD.HiIndx.Quench

20.5.64 SC.Quench.Outage [L1,2,3,4,5]

Brief Definition	SC.Quench.Outage gives the time after quench is initiated when the sensing array becomes reliably sensitive to newly incident photons.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	For Geiger mode operation, the array is quenched to make it receptive to new photons. This requires changing the bias on the detector elements and giving them time to settle into a receptive state. During this period, the sensor is not sensitive to incident photon flux. Of course, after an avalanche is triggered and before a quench, the sensing element is also blind to additional incident photons. The purpose of multiple quenches is to reduce the portion of the receive window when the detectors are blind. However, there is a cost and SC.Quench.Outage quantifies that. Since there is a blind interval after a quench, when multiple quenches are used per receive window, the outage time for each quench determines what ranges a particular pulse is blind to. This information is valuable for interpreting the measured photon pattern. SC.Quench.Outage is the number of fast data capture clock counts after the quench commences when the detector array is in a receptive state again. The fast data capture clock interval is the interval of time granularity at which the APD latching instant is quantized when it is captured in a reading indicating the firing of a detector element, presumably recording the time of photon arrival.
Governing quality metrics	Correct value populated. While this number may be element dependent, we need to provide a number that envelopes the response of the entire detector array.
See Also	SC.Quench.Times(j)

[234] This bases the Quench timing at the start of the IPP. It seems that the more interesting interval is the time since start of the receive window. But that drives negative times for the first quench. So we regularize the values with this convention. The IPP receive window start time is available in SC.Pulse.RecordStrt.*TileNum*(ip, j).

SC.RgAmb.*

SC.RgAmb.* parameters describe sensor configuration items related to the use of PRFs sufficiently high that at the range to the scene, there are multiple pulses in transit at any time and consequently pulses returning from multiple ranges are being simultaneously received.

20.5.65 SC.RgAmb.Num.*Il*(isnc,j) [L1,2,3,4,5]

Brief Definition

SC.RgAmb.Num.*Il*(isnc,j) gives the range ambiguity number by illuminator for each scan nadir class. The range ambiguity number is one less than the number of pulses that can be simultaneously incident on the receiver after reflecting at different ranges. Ambiguity numbers greater than zero result from IPP times being less than round trip times to the most distant visible scatterer.

indices

"*Il*" while part of the name of the data item, acts as an index. The value of *Il* is the alpha-numeric translation of the illuminator index and ranges from the BCS translation of PD.LoIndex.Laser(1) to PD.HiIndx.Laser(1), that is, over the laser illuminators in the lidar used for this collection.
This provides channel dependence for the ambiguity number so that range gates can spatially overlap for channels with different PRFs.

Index, isnc, ranges from PD.LoIndx.RgAmb(1) to PD.HiIndx.RgAmb(1), that is, generally from 1 to the number of unique scan phase dependencies (scan nadir classes) for the ambiguity number. **Note well**: while the scan nadir class is is just an index and is not a value supported by its own metadata item, all places that use this index must consistently align the meanings of the value of this index. SC.Scan.NomNadir.* also use an index that must follow the same sequence of scan nadir class as used as a reference here.

Index, j, ranges from PD.LoIndx.RgAmb(2) to PD.HiIndx.RgAmb(2), that is generally from 1 to the number of range ambiguity values for the scan nadir class(es) with the highest count of ambiguity numbers used. See the discussion in SC.RgAmb.StratFlg(j).

Units/representation

Dimensionless, UInt16

Treatment/usage

<u>Ambiguity number discussion</u>
The ambiguity number is necessary for correct range calculation. See the figure and explanation in the paragraph detailing SC.Pulse.TxDelay.*TileNum*(ip). Without valid values for the range ambiguity number, the range to an XYZ point could be off by a multiple of the distance

light travels in an IPP. The ambiguity number could also support proper interpretation of range ambiguous content in the (raw) point cloud.
The range ambiguity number is the number of IPPs that occur between the transmission of a pulse and the IPP in which the return of that pulse from the intended scene range swath is expected. For ranges so close that a pulse echo is received in the same IPP as the transmission, the range ambiguity number is zero. Thus $N_{amb} = N_{IPP_receive} - N_{IPP_transmit}$ where $N_{IPP_receive}$ is the number of the IPP the return is received in due to the pulse transmitted in the IPP number given by $N_{IPP_transmit}$.

<u>Using SC.RgAmb.Num.*Il*(isnc,j) to direct a processor to select the intended ambiguity number for an IPP</u>
We can only provide enterprise L-1 ambiguity number data based on the planning of the collection. As discussed in the paragraph treating SC.RgAmb.StratFlg(j), it is possible that the planning process was not detailed enough to assure that the supplied ambiguity numbers for range calculation are all correct.

SC.RgAmb.Num.*Il*(isnc,j) is used by the application that converts time of arrival data into round trip times, which might then be used to determine XYZ positions for lidar returns. In order to determine the ambiguity number to be used for a particular IPP, one first must determine the scan nadir class for the scan containing that IPP by consulting SC.Scan.NadirClass.*ChNum*(is) and SC.Pulse.Scan.Num.*TileNum*(ip, j). One then consults SC.Pulse.Scan.Phase.*TileNum*(ip, j) to obtain the scan phase for the IPP. Then one determines which ambiguity number unique scan phase bin the IPP falls into by checking SC.Scan.PhaseBins(isnc,j). Finally, the scan nadir class index and the scan phase bin index are used in SC.RgAmb.Num.*Il*(isnc,j) to look up the IPP specific ambiguity number.

<u>Why SC.RgAmb.Num.*Il*(isnc,j) provides an array of ambiguity numbers</u>
The assignment of higher than unity value to PD.HiIndx.RgAmb(1) and PD.HiIndx.RgAmb(2), is discussed in the paragraph for SC.RgAmb.StratFlg(j). If there is only a single scan nadir class in a data set, then PD.LoIndx.RgAmb(1) will be set to PD.HiIndx.RgAmb(1) providing a range of 1 value. If the number of ambiguity values for the only scan nadir class is 1, then PD.LoIndx.RgAmb(2) will be set to the same value as PD.HiIndx.RgAmb(2). The SC.RgAmb.Num.*Il*(isnc,j) array then reduces to a single ambiguity number value.

Governing quality metrics

The validity of this number is supplied in SC.RgAmb.StratFlg(2)

See Also SC.RgAmb.StratFlg(j) SC.Scan.NadirClass.*ChNum*(j) PD.HiIndx.RgAmb(j) SC.Pulse.Scan.Phase.*TileNum*(ip, j) SC.Scan.PhaseBins(isnc,j)

20.5.66 SC.RgAmb.StratFlg(j) [L1,2,3,4,5]

Brief Definition SC.RgAmb.StratFlg(j) carries indicators of the range ambiguity management strategy used in the collection.

indices Index j=1,2

Units/representation Dimensionless code value, UInt8

Treatment/usage While range ambiguity management can be done so as to lead to a single range ambiguity number for a single collection operation, a single ambiguity number for an entire collect is not always capable of supporting the most efficient data recording or highest power limited area rate. Below we identify the options for range ambiguity management that can be supported by the CMMD metadata. These lead to the possibility of a single collect consisting of multiple scan nadir classes, each of which might be used with more than one range ambiguity value each.

SC.RgAmb.StratFlg(1) carries a code that indicates PRF/ ambiguity number variation strategy given in the following table.

Code	**PRF**	**Namb**	**Gate delay**	range swath **margin**
1	breathing	fixed	fixed	optional
2	breathing	fixed	varying	optional
3	constant	varying	fixed	optional
4	constant	fixed	varying	optional
5	constant	fixed	fixed	needed

SC.RgAmb.StratFlg(2) carries a code that indicates confidence that the populated range ambiguity data in SC.RgAmb.Num.*Il*(isnc,j) is precisely correct vs nominally correct

Code= 1—> high fidelity population of ambiguity number. Low chance that return positions could be off by a full range swath or more in range.

Code= 2—> Low fidelity population of ambiguity number. Likely need to use continuity and collateral data to smooth out XYZ base.

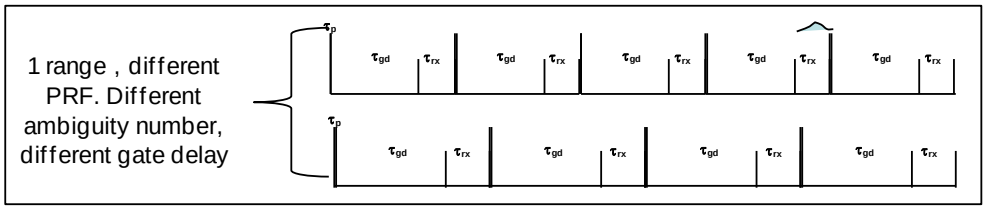
The ambiguity number can be interpreted in terms of the number of pulses "in the air"[235] simultaneously. For a given PRF, there will be more pulses in transit for longer ranges and fewer pulses in transit at shorter ranges. Since the timing of returns is ambiguous if the single IPP receiving window duration encompasses returns from the same range swath by more than one pulse, it will be normal to listen for only the (two way) light travel time across the range swath to be mapped, and for that to be less than the time between pulses. This effectively means that there is a single receiving window for each pulse transmitted. Still, the receiver at a single instant could have photons incident from multiple ranges. This leads to a point cloud that has a superposition of scenes in it. The intent is that only one of those have the dominant energy and that we can compute the range to points in that scene very accurately.

Depending on the intended area coverage rates, sensor designs and collection modes may be choosing to manage range ambiguous collection in different ways. At the highest level, the issue can be considered one of time variation of the range ambiguity number. Simple, single range collection can easily[236] work with only one range ambiguity number for an entire collection at the swath level. An additional strategy could be scans that have a single (i.e., constant) range ambiguity number value, but possibly different values for different scans. Lastly, scans could cover enough range variation that the ambiguity number varies within the scan. While the latter case may be complex to control, if it enables meeting a capability or performance requirement, the metadata definitions should not be permitted to obstruct the selection of that strategy.

Review of Ambiguity management strategies

Generally, there are multiple combinations of PRF, ambiguity number and gate delay to collect any specific range of target. This is illustrated in the example below which places a receive window at the same time after a transmitted pulse with two different IPP structures.

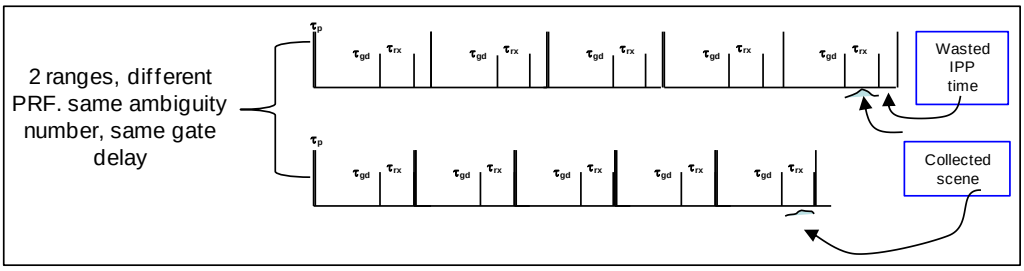
[235] Quotes used to indicate that this is a shorthand /figurative expression. Atmosphere is not necessary for ambiguity. The ray path may encounter multiple media and still support photon propagation.
[236] This is a virtue of constant nadir angle scanning.



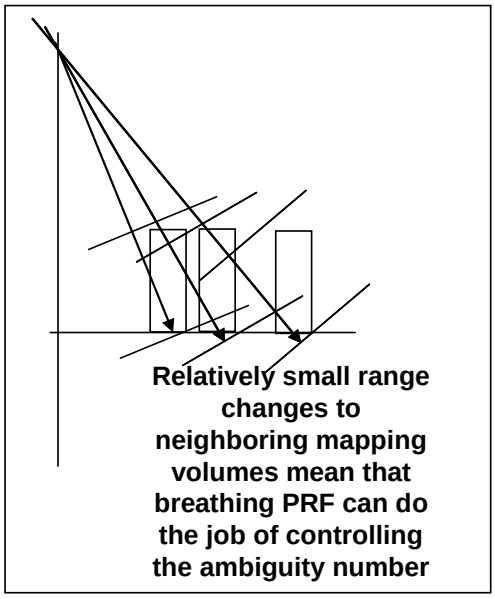
Notation

τ_p	Pulse duration
τ_{gd}	Gate delay duration
τ_{rx}	Receive window duration

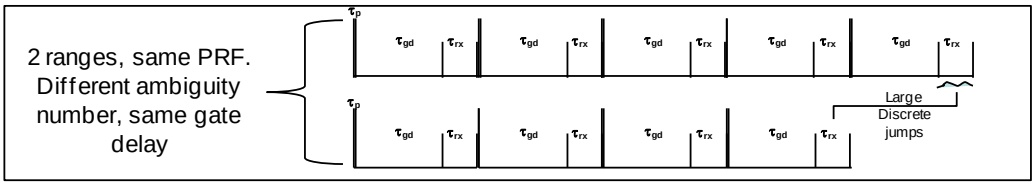
When directing the lidar to different ranges while holding the ambiguity number constant, the PRF changes. Over cycles of range variation the PRF is said to breathe with the ambiguity number held constant. This is illustrated in the following two (code=1) diagrams.



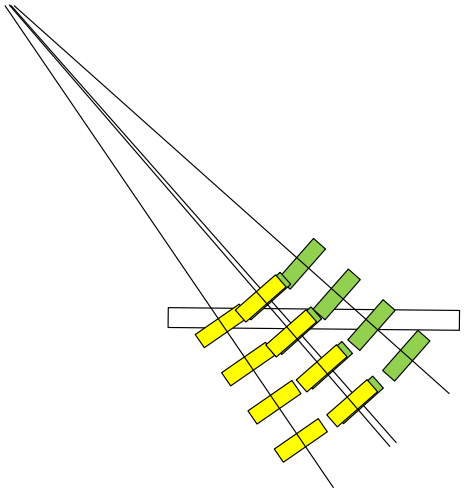
Below is a spatial diagram of the same principle. When the range changes are small, the wasted IPP time may be small enough to avoid the need to manage it.



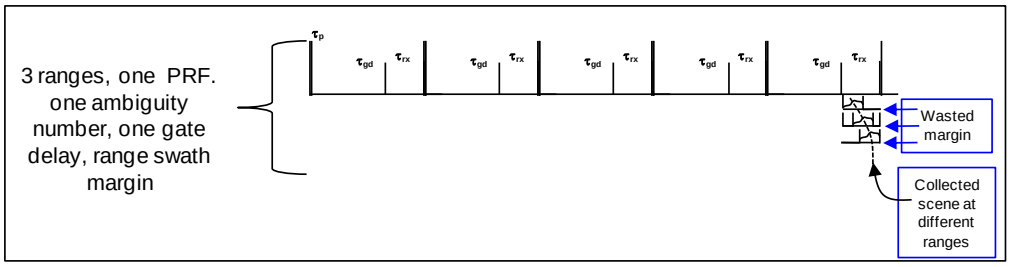
Ambiguity number changing leads to coarse range control unless the ambiguity number is extremely large. See the (code=3) illustration below.



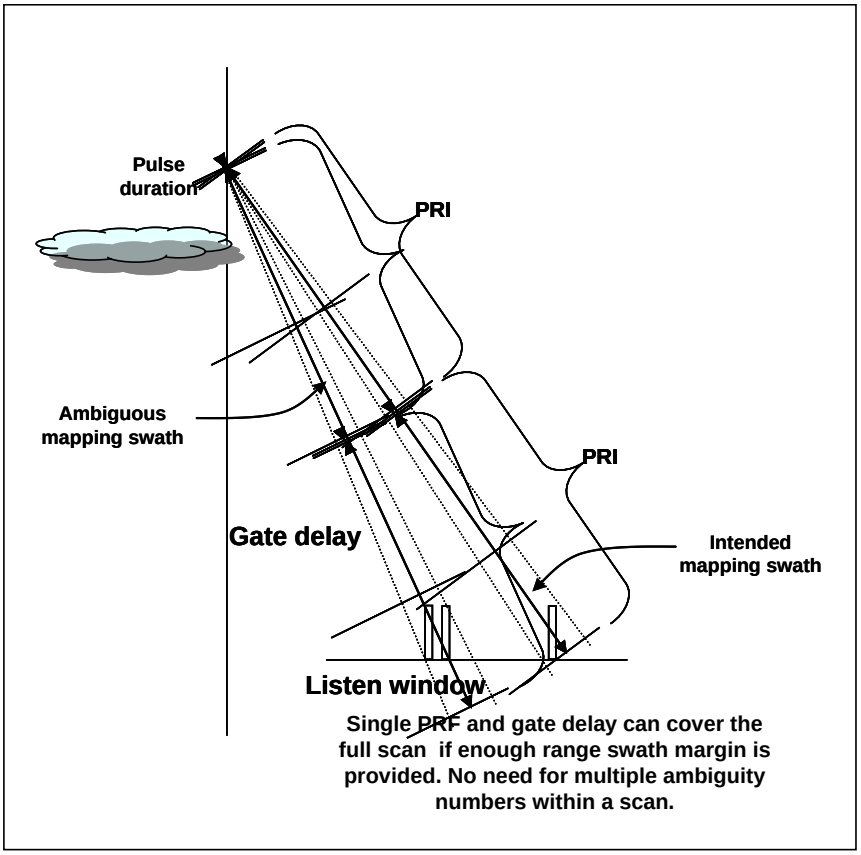
This could be a desirable property. For example, along track scanning to collect aspect diversity in a single collection could use one ambiguity number for the closer to nadir angle and then use a larger ambiguity number for the more forward looking part of the scan. This would be necessary for the more forward looking pulses to cover the earth's surface.



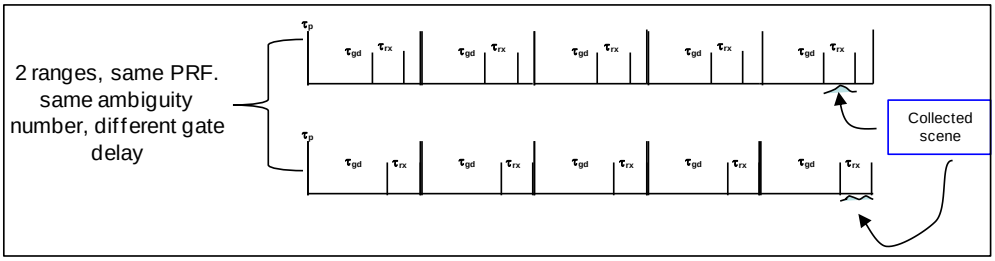
A simple IPP control structure can work, but it results in inefficient collection. This wastes data rate. See below. (code=5)



A spatial view of this strategy is show below.

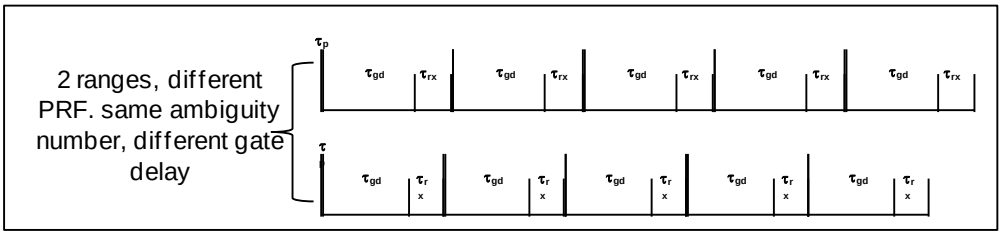


Not controlling enough of the IPP structure parameters can lead to inefficiency at certain scan angles. See below (code=4).



While appealingly simple, the above strategies with limited IPP structure parameter variation may not be consistent with achieving area rate requirements. The principle of not permitting the metadata design to limit sensor collection optimization strategy options drives us to be more general.

More efficient collection requires more control of the IPP structure. The following (code=2) illustration shows this.



To deal with the possibilities of ambiguity number variation from no variation, to scan specific and even variation within scans, the SC.RgAmb.Num.*Il*(isnc,j) array that assigns ambiguity numbers to IPPs has been made into a small but variable 2-D array. The rows, of which there can be one or many, cover a single scan nadir class each. A scan nadir class is a collection specific bracketing of range ambiguity behavior. The scans, being periodic repeating beam maneuvers, only come in a few such classes for any collection, although the scan relative geometry (sensor to covered scene) is likely repeated up to hundreds, if not thousands of times each during collection of a swath. A single class might be determined by the combination of the choice of scan direction and range of off nadir angle value for its scans. Thus, a given scan has a variation of the ambiguity number with scan phase that is documented via its scan nadir class index value. For example, a collection could employ 3 scan Nadir classes. The scans that are closest to nadir might all have the same, constant range ambiguity number. The second class might range just far enough off nadir that the option to permit 2 different ambiguity numbers to be used for each of those scans needs to be exercised. In principle a third, further from nadir, class of scans might be executed such that 3 ambiguity numbers arise in each scan. To describe the phase within the scan at which the ambiguity numbers jump (range ambiguity numbers are only integer valued), the second index of SC.RgAmb.Num.*Il*(isnc,j) is available. In the common operational case that the ambiguity number remains constant during all scans, the Lo and Hi Index value for that dimension of the array can both be set to 1. By the same token, if the mode is designed with all scans having the same ambiguity number, the Lo and Hi Index for the first dimension of SC.RgAmb.Num.*Il*(isnc,j) can be set to 1 as well.

The motivation to have multiple lidar interrogations of single scene (voxel) positions incentivizes the mode design toward higher PRFs, that is, potentially higher ambiguity numbers. Increasing the dwell of the beam in the direction of

a single scene point also supports higher numbers of interrogations, but at the cost of area rate or beam (that is, instantaneous FOV) width.

If some regions on the ground will only be interrogated from longer ranges than the nominal range or average range during a collection, it could be advantageous to have more interrogations of the longer range scenes. This can happen by increasing the dwell (slowing the scan) or by increasing the PRF. The latter leads to a double impact, increasing the ambiguities from both a higher PRF and increased path length, when each alone leads to increased ambiguity number. It is highly unlikely the lidar mode design would do that.

The ambiguity number is an essential part of the round trip time calculation for returns from every transmitted pulse. If it is not directly provided, it must be calculable from given data. Assumptions needed for calculating the ambiguity number are the approximate terrain ECEF extent, sensor position and orientation and PRF. Making a low level DTED available as metadata is a bit onerous just to avoid providing a few ambiguity numbers. A good question is whether the mission planning uses the approximate ground elevation in selecting the flight line and/or the PRF. The PRF could be selected only by a scan rate and oversampling criterion. Supplying a flight line height above ground data item along with an flat/ellipsoidal surface model and nominal off nadir direction could permit ambiguity number calculation that would be good most of the time. It could fail when terrain was significant like valleys, hill sides etc, especially depending on range swath length. Another approach is to accept the ambiguity wrap in the initial XYZ derivation from time and angle. The impact of assuming the wrong ambiguity number could be fixed in later processing by forcing continuity, reconciling with other sources and a'priori DEMs etc. That would make the flat earth assumption just fine for stage-0 to stage-1 processing to create "raw" point clouds. One could argue that is the right way to go. Supplied ambiguity numbers could be wrong [they could have been generated in the field from a flat earth or sea level assumption, for example] and the correction processing mentioned above would be needed to fix up the impacts of the bad metadata. So we support documenting the range ambiguity variation strategy when it can be populated in SC.RgAmb.StratFlg(1), and saying how it was intended to be used. SC.RgAmb.StratFlg(2) indicates whether the values are high or low confidence so that one could anticipate the benefits of the extra processing pass to clean up discontnuities arising from locally incorrect ambiguity number values.

Governing quality metrics

See Also SC.RgAmb.Num.*Il*(isnc,j)

SC.Scan.*

SC.Scan.* parameters describe sensor configuration items related to the periodic beam steering maneuvers called scans.

20.5.67 SC.Scan.Axes.ThetaMaj(iscp, ich) [L1,2]

Brief Definition	SC.Scan.Axes.ThetaMaj(iscp, ich) gives the extent of a scanning maneuver of the optical axis in angle space relative to the sensor for each scan cone parameterization set used in the data set.
indices	Index, "iscp" , ranges from PD.LoIndx.Scan(2) to PD.HiIndx.Scan(2), that is over the **s**can **c**one **p**arameterization sets used in the data set Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. This reiterates the issue identified in 10.1.3.1 where the relationship of channel specific scan parameters and channel documentation was discussed. If a data set is to have scan parameters provided, and the collection scan parameterization was channel specific, that data set would need to provide channel definitions back to those which were collected.
Units/representation	Degrees, float
Treatment/usage	See appendix 10.1.2.4 discussion of scan parameterization. Scans have been generalized to be periodically repeated beam maneuvers on an elliptical cone in angle space relative to the sensor. The major axis of this elliptical cone is centered on the scan cone axis and gives the angular sweep extent of the scan parameterization, iscp, in one direction. SC.Scan.Axes.ThetaMaj(iscp,ich) is the angular size of the elliptical scan cone for channel "ich" of the iscpth scan cone parameterization. When scans are linear, the minor axis will have size zero and the length of the scan trace on a flat ground facet will be determined by SC.Scan.Axes.ThetaMaj(iscp,ich) and the collection range. The case of along track scans that have the same ground extent for different track offsets (that is, nominal nadir values of the scan) requires that the scan elliptical cone parameters be separately valued for different scan axis directions. Thus SC.Scan.Axes.ThetaMaj(iscp,ich), which is indexed by the scan cone parameterization set value, is able to describe each of the scan configurations used in a collection.

Governing quality metrics	
See Also	PD.HiIndx.Scan PD.LoIndx.Channlzn.ChRg PD.LoIndx.Channlzn.Rg_m

20.5.68 SC.Scan.Axes.ThetaMin(iscp,ich) [L1,2]

Brief Definition	SC.Scan.Axes.ThetaMin(iscp,ich) gives the extent of a scanning maneuver of the optical axis in angle space relative to the sensor and permits curvature to the scan pattern.
indices	Index, "iscp" , ranges from PD.LoIndx.Scan(2) to PD.HiIndx.Scan(2), that is over the **s**can **c**one **p**arameterization sets used in the data set Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. This reiterates the issue identified in 10.1.3.1 where the relationship of channel specific scan parameters and channel documentation was discussed. If a data set is to have scan parameters provided, and the collection scan parameterization was channel specific, that data set would need to provide channel definitions back to those which were collected.
Units/representation	Degrees, float
Treatment/usage	See appendix 10.1.2.4 discussion of scan parameterization. Scans have been generalized to be periodically repeated beam maneuvers on an elliptical cone in angle space relative to the sensor. The minor axis of this elliptical cone is centered on the scan cone axis and gives the angular sweep extent of the scan cone parameterization, iscp, in one direction. SC.Scan.Axes.ThetaMin(iscp, ich) is the angular size of the elliptical scan cone minor axis for channel ich of the iscpth scan cone parameterization. When scans are linear, the minor axis will have size zero [SC.Scan.Axes.ThetaMin(iscp,ich)=0] and the length of the scan trace on a flat ground facet will be determined by SC.Scan.Axes.ThetaMaj(iscp,ich) and the collection range. The case of along track[237] scans that have the same ground extent for different track offsets (that is, nominal nadir values of the scan) requires that the scan elliptical cone parameters be separately valued for different scan axis directions. Thus,

[237] While a purely track parallel scan for a straight platform trajectory might not have the curvature provided by a non-zero minor axis, there can be scan arcs that are oriented *generally* along track and that *could* utilize a non-zero minor axis for the scan cone.

	SC.Scan.Axes.ThetaMin(iscp,ich), which is indexed by the scan cone parameterization set value, is indexed so as to be able to describe each of the scan configurations used in a collection.
Governing quality metrics	
See Also	PD.HiIndx.Scan(j) PD.LoIndx.Channlzn.ChRg_ PD.HiIndx.Channlzn.Rg_m SC.Scan.Axes.ThetaMaj(iscp, ich)

20.5.69 SC.Scan.ConePulseTable(isci) [L1,2]

Brief Definition	SC.Scan.ConePulseTable(isci) lists the pulse number of the first pulse of each separate **s**can **c**one parameterization set **i**nvocation.
indices	Index, "isci" , ranges from PD.LoIndx.Scan(4) to PD.HiIndx.Scan(4), that is over **s**can **c**one **p**arameterization set invocations used in the data set.
Units/representation	Dimensionless, UInt32
Treatment/usage	Per the scan parameterization discussion in 10.1.2.4, scan geometry relative to the sensor frame is described by an elliptical cone whose parameters are the direction of the cone axis and the major and minor axes of the ellipse formed by a plane cutting the cone perpendicularly to the cone axis. While it is permissible to have all the scans in a collect use the identical cone axis direction [typical of a whiskbroom scanning mode], the metadata must also support the possibility of sequences of blocks of scans with different scan cone axis directions for each block of the sequence but the same cone axis direction for scans within a single block. It is then possible for the collection to repeat these sequences. It is possible that each distinct cone parameterizaton in such a sequence could lead to a separate value of the scan nadir class, which acts as a key for ambiguity number data and nadir angle parameters of the individual scans. SC.Scan.ConePulseTable(isci) gives the (global) IPP[238] number of the first IPP of an invoked scan cone parameterization. Any subsequent cone parameter changes go into the list sequentially. The index of this data vector is "isci", which ranges over the number of the sequential scan cone parameterization choice invocations within a data set. When all of the scans in a data set have the same scan cone parameterization in the sensor frame, the length of the SC.Scan.ConePulseTable is 1 and PD.LoIndx.Scan(4) and PD.HiIndx.Scan(4) are both set to the same value. The rule to interpreting the table is always that all scans occurring during pulses on or after a pulse

[238] Global IPPs are indexed starting at 0 as the first IPP in a collection and the index increases to one less than the total number of IPPs in the collection. This is not a tiled table and no tile based IPP counting is done here.

number supplied in the SC.Scan.ConePulseTable(isci) and before the next pulse number supplied use the same scan cone parameterization set[239]. In particular, all scans covering pulses after the last entry in SC.Scan.ConePulsetable(isci) use the last cone parameterization set's data.

Governing quality metrics

See Also PD.HiIndx.Scan(j)

20.5.70 SC.Scan.N_Major(iscp,ich,k) [L1,2]

Brief Definition SC.Scan.N_Major(iscp,ich,k) gives the orientation of the scan major axis for each choice of scan cone axis.

indices Index, iscp, ranges from PD.LoIndx.Scan(2) to PD.HiIndx.Scan(2), that is from 1 to the number scan cone parameterizations. This is the same index as "iscp" in the SC.Scan.U_Cone(iscp, ich,k) parameter and these two data items constitute a scan cone description table with a row for each distinct set of sensor frame parameters.

Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. This reiterates the issue identified in 10.1.3.1 where the relationship of channel specific scan parameters and channel documentation was discussed. If a data set is to have scan parameters provided, and the collection scan parameterization was channel specific, that data set would need to provide channel definitions back to those which were collected.

Index, k=1,2 for direction cosines measured from the sensor x and y axes respectively.

Units/representation Dimensionless, float

Treatment/usage SC.Scan.N_Major(iscp,ich,k) gives the normal to the plane of the scan cone major axis relative to the sensor frame for each choice of scan parameterization index, "iscp" and for each channel, "ich". Refer to the appendix figure 10.1.2-3. When the scan cone has minor axis equal to zero, the scan is in a plane relative to the sensor frame and the direction of a scan

[239] That is a set of more than one cone specification per starting pulse number when there are channel specific scan parameter values.

defined as the intersection of the scan plane and a ground tangent facet can be computed [with the auxiliary assumption that the sensor frame has no acceleration in the ECEF frame]. In the general case, the scan cone is not necessarily flattened to a plane and the plane defined here gives the orientation of any cone asymmetry in terms of the major axis direction of the cone. Scan tracks on the ground can be computed from this scan axis information, but in the general case they are complicated by the linear and angular acceleration of the sensor frame.

We use direction cosines of angles α_{major} and β_{major} from the diagram to specify a unit vector in the sensor frame that is normal to the plane containing the major axis of the elliptical cone and the vertex of the cone. These values populate the k=1,2 elements respectively for each value of iscp and ich.

Note that while it is possible that the scan cone index changes in lock step with the scan nadir class, it is not necessary and so the index symbols "isnc" and "iscp" [240]are both used to support the general case, although common use might actually be that these two indices are equivalent.

Governing quality metrics: This item should have the same sequence of row index values as the data for the scan cone axis direction. Locking these indices together forms a table with consistent data on each row. The uncertainty in the values of the direction cosines is the percentage of error in the data item.

See Also: SC.Scan.U_Cone(iscp, ich,k) SC.Scan.ConePulseTable(isci) PD.HiIndx.Scan(j) PD.LoIndx.Channlzn.ChRg_ PD.HiIndx.Channlzn.Rg_m

20.5.71 SC.Scan.NomGroundLen(iscp) [L1,2,3,4,5]

Brief Definition: SC.Scan.NomGroundLen(iscp) gives the nominal length of a scan as projected to the ground surface from the sensor.

indices: Index, iscp, ranges from PD.LoIndx.Scan(2) to PD.HiIndx.Scan(2), that is from 1 to the number scan cone parameterizations for a collection. This is the same index as iscp in the SC.Scan.U_Cone(iscp, ich,k) parameter and these two data items constitute a scan cone description table with a row for each distinct set of sensor frame parameters.

A channel index is not needed here because the nominal length of all scans on the ground is channel independent.

Units/representation: meters, float

Treatment/usage: SC.Scan.NomGroundLen(iscp) gives the nominal length of a scan cone

[240] In other words, the iscp index here is logically decoupled from the scan nadir class, but can be effectively equivalent due to populated values when appropriate.

segment as projected to the nominal ground surface from the sensor. This is the length of the segment of ground collected by the lidar in the iscpth scan cone parameterization. Nominal sensor elevation (such as the average over the interval of collection for the data set) must be assumed[241] in this calculation as well.

Governing quality metrics: [nominal scan length -actual average scan length]/ actual average scan length

See Also: SC.Scan.U_Cone(iscp, ich,k)

20.5.72 SC.Scan.NadirClass.*ChNum*(is) [L1,2]

Brief Definition: SC.Scan.NadirClass.*ChNum*(is) gives the value of the scan nadir class for each scan by channel.

indices:

Default case:

Index, is, ranges from PD.LoIndx.Scan(3) to PD.HiIndx.Scan(3), that is, over the number of scans in the data set.

Repeated data Redundancy reduction case:

The SC.Scan.NadirClass.*ChNum*(is) becomes SC.Scan.NadirClass.*ChNum*(j), where the range of j is from 1 to PD.IntTileMap.NadirClass.*ChNum*.Flg [or 0: (PD.IntTileMap.NadirClass.*ChNum*.Flg-1), as coding practice may prefer]. There is one row in SC.Scan.NadirClass.*ChNum*(j) (viewed as a column vector) for each nadir class value [given by collection channel as determined by *ChNum*] that is repeated consecutively the number of times indicated by PD.IntTileMap.NadirClass.*ChNum*.Reps(j).

Evaluated substring, *ChNum*, acts as the logical equivalent of a channel index, ich, which ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. This reiterates the issue identified in 10.1.3.1 where the relationship of channel specific scan parameters and channel documentation was discussed. If a data set is to have scan parameters provided, and the collection scan parameterization was channel specific, that data set would need to provide channel definitions back to those which were collected.

Units/representation: Dimensionless, UInt8

Treatment/usage: A scan nadir class is a classification of a scan by its range ambiguity value pattern. All members of a “scan nadir class” have the same progression of

[241] The geometry for projection of the scan cone to the ground is determined by the sensor height above ground and the cone axis direction in the earth frame. This axis is given as a scan parameter in the sensor frame.

ambiguity number with scan phase. The simplest collection mode designs have only one scan nadir class and all the scans in the collect belong to it. Furthermore, in this simplest case, there is only one ambiguity number value for all IPPs in all of the scans. More complicated designs have multiple scan nadir classes. In the case that different scan nadir classes exist, but have single ambiguity numbers, each scan nadir class will have its own unique ambiguity number. In the most complicated scanning schemes, the scan nadir classes have different numbers of ambiguity numbers that are intended to be used as the scan phase increments through the scan.

See discussion of scans in 10.1.2.4 and 10.1.3.2. Due to the combined effects of the nominal nadir angle for a scan, the variation in range to the scene due to the variation of nadir angle, the scan direction and range ambiguity management strategy, each scan belongs to one of a few nadir classes uniquely identified by the value and value range of the ambiguity number over the scan. Scans, being periodically repeating optical axis steering maneuvers, exhibit only one or a few distinct cases of relative geometry history between the optical axis and sensor body in a single collection. Generally each case may have thousands of instantiations during a single swath collection operation. Each individual scan will be an instantiation of one of the scan nadir classes. The value of SC.Scan.NadirClass.*ChNum*(is) is the numeric identifier of the unique range ambiguity value pattern (which we term a "scan nadir class")[242] that charactizes the isth scan in channel "ich". The scan nadir class is just an index (indicated by "isnc" when it appears in the properties of scan descriptor arrays SC.Scan.NomNadir.*(isnc) and SC.RgAmb.Num.*Il*(isnc,j) and is not a value supported by its own metadata item. **This data definition construct requires that all places that use this index must consistently align the meanings of the value of this index.** SC.Scan.NomNadir.* also uses the isnc index so population of SC.Scan.NomNadir.* must follow the same sequence of scan nadir class as used as a reference here. The value of SC.Scan.NadirClass.*ChNum*(is) ranges from PD.LoIndx.RgAmb(1) to PD.HiIndx.RgAmb(1).

We use the *ChNum* construct to assure that we can avoid redundancy due to repeated values of scan nadir class in one or more consecutive runs of scans without forcing the channel specific nadir classes to change in lock step. A common situation will be that many consecutive scans will be in the same

[242] A scan nadir class has a particular Namb variation pattern within each single scan within that nadir class. The pattern may be as simple as no variation of the ambiguity number within a scan or it may be a short sequence of particular values of the ambiguity number.

scan nadir class. In that case the size of this array can be substantially reduced from the number of scans in the product to only as many different scan nadir class starts as there are across the scans. This is enabled through auxiliary use of PD.IntTileMap.NadirClass.*ChNum*.Reps(j) and PD.IntTileMap.NadirClass.*ChNum*.Flg

Governing quality metrics

See Also SC.RgAmb.Num.*Il*(isnc,j) PD.IntTileMap.NadirClass.*ChNum*.Flg SC.Scan.NomNadir.Mean(isnc) PD.IntTileMap.NadirClass.*ChNum*.Reps(j) PD.HiIndx.Scan(j) PD.LoIndx.RgAmb(j)

20.5.73 SC.Scan.NomNadir.Max(isnc) [L1,2]

Brief Definition SC.Scan.NomNadir.Max(isnc) gives the nominal maximum optical axis angle from nadir collected for scans in each ambiguity number pattern unique scan nadir class.

indices Index, isnc, ranges from PD.LoIndx.RgAmb(1) to PD.HiIndx.RgAmb(1), that is generally from 1 to the number of unique scan phase dependencies (one phase dependency per scan nadir class) for the ambiguity number. **Note well**: while the scan nadir class is just an index and is not a value supported by its own metadata item, all places that use this index must consistently align the meanings of the value of this index. SC.Scan.NomNadir.* also use an index that must follow the same sequence of scan nadir class as used as a reference here.

Units/representation Degrees, float

Treatment/usage Consider the value of the maximum angle measured from nadir to the optical axis direction over a scan. This is determined by the scan cone parameters and the sensor frame orientation. Over the variation of the sensor frame orientation during a collect, the impact of the scan cones moving about in the earth frame is to introduce some variation of the maximum nadir angles. Also, in the absence of sensor frame orientation variations, there could be multiple scan cone parameterizations in use that all have the same ambiguity number and thus belong in the same scan nadir class. SC.Scan.NomNadir.Max(isnc) is a representative value of maximum angle measured from nadir to the optical axis direction over the scans in a scan nadir class. A value is supplied for each scan nadir class, that is, each distinct ambiguity number pattern with respect to scan phase. So, if there are several scan parameterizations that share a single ambiguity number, SC.Scan.NomNadir.Max(isnc) is the average of the maximum nadir angle values over those parameterizations. The maximum nadir angle for a given scan cone parameterization is derivable from the scan cone parameters and the nominal sensor frame orientation.

Governing quality metrics Being a nominal nadir maximum, this need only be a value that is

representative, not exact. Values must be populated such that the cosine of the nadir angle is within 1% of the ideal/error-free value.

See Also PD.HiIndx.RgAmb(j)

20.5.74 SC.Scan.NomNadir.Mean(isnc) [L1,2,3,4,5]

Brief Definition SC.Scan.NomNadir.Mean(isnc) gives the nominal mean optical axis angle from nadir collected for scans in each ambiguity number pattern unique scan nadir class.

indices Index, isnc, ranges from PD.LoIndx.RgAmb(1) to PD.HiIndx.RgAmb(1), that is generally from 1 to the number of unique scan phase dependencies (one phase dependency per scan nadir class) for the ambiguity number. **Note well**: while the scan nadir class is just an index and is not a value supported by its own metadata item, all places that use this index must consistently align the meanings of the value of this index. SC.Scan.NomNadir.* also use an index that must follow the same sequence of scan nadir class as used as a reference here.

Units/representation Degrees, float

Treatment/usage Consider the average value of the angle measured from nadir to the optical axis direction over a scan. This is determined by the scan cone parameters and the sensor frame orientation. Over the variation of the sensor frame orientation during a collect, the impact of the scan cones moving about in the earth frame is to introduce some variation of the nadir angles. Also, in the absence of sensor frame orientation variations, there could be multiple scan cone parameterizations in use that all have the same ambiguity number and thus belong in the same scan nadir class. SC.Scan.NomNadir.Mean(isnc) is a representative value of the angle measured from nadir to the optical axis direction over the scans in a scan nadir class. A value is supplied for each scan nadir class, that is, each distinct ambiguity number pattern with respect to scan phase. So, if there are several scan parameterizations that share a single ambiguity number, SC.Scan.NomNadir.Mean(isnc) is the average of the mean nadir angle values over those parameterizations. The mean nadir angle for a given scan cone parameterization is derivable from the scan cone parameters and the nominal sensor frame orientation. Thus, SC.Scan.NomNadir.Mean(isnc) is not necessarily the same as the mean for a single cone parameterization.

Governing quality metrics Being a nominal nadir mean, this need be only a value that is representative, not exact. Values must be populated such that the cosine of the nadir angle is within 1% of the ideal/error-free value.

See Also

20.5.75 SC.Scan.NomNadir.Min(isnc) [L1,2,3,4,5]

Brief Definition	SC.Scan.NomNadir.Min(isnc) gives the nominal minimum optical axis angle from nadir collected for a scan in each ambiguity number pattern unique scan nadir class.
indices	Index, isnc, ranges from PD.LoIndx.RgAmb(1) to PD.HiIndx.RgAmb(1), that is generally from 1 to the number of unique scan phase dependencies (one phase dependency per scan nadir class) for the ambiguity number. **Note well**: while the scan nadir class is just an index and is not a value supported by its own metadata item, all places that use this index must consistently align the meanings of the value of this index. SC.Scan.NomNadir.* also use an index that must follow the same sequence of scan nadir class as used as a reference here.
Units/representation	Degrees, float
Treatment/usage	Consider the minimum value of the angle measured from nadir to the optical axis direction over a scan. This is determined by the scan cone parameters and the sensor frame orientation. Over the variation of the sensor frame orientation during a collect, the impact of the scan cones moving about in the earth frame is to introduce some variation of the minimum nadir angles. Also, in the absence of sensor frame orientation variations, there could be multiple scan cone parameterizations in use that all have the same ambiguity number and thus belong in the same scan nadir class. SC.Scan.NomNadir.Min(isnc) is a representative value of the minimum angle measured from nadir to the optical axis direction over the scans in a scan nadir class. A value is supplied for each scan nadir class, that is, each distinct ambiguity number pattern with respect to scan phase. So, if there are several scan parameterizations that share a single ambiguity number, SC.Scan.NomNadir.Min(isnc) is the average of the minimum nadir angle values over those parameterizations. The minimum nadir angle for a given scan cone parameterization is derivable from the scan cone parameters and the nominal sensor frame orientation. Thus, SC.Scan.NomNadir.Min(isnc) is not necessarily the same as the minimum for a single cone parameterization.
Governing quality metrics	Being a nominal nadir maximum, this need only be a value that is representative, not exact. Values must be populated such that the cosine of the nadir angle is within 1% of the ideal/error-free value.
See Also	

20.5.76 SC.Scan.PhaseBins(isnc,j) [L1,2,3]

Brief Definition	SC.Scan.PhaseBins(isnc,j) gives the upper bound of scan phase for each ambiguity number specific phase range of a scan for a given scan nadir class.
indices	Index, isnc, ranges from PD.LoIndx.RgAmb(1) to PD.HiIndx.RgAmb(1), that is, generally from 1 to the number of unique scan phase dependencies for the

ambiguity number (these are called scan nadir classes in this document). **Note well**: while the scan nadir class is just an index and is not a value supported by its own metadata item, all places that use this index must consistently align the meanings of the value of this index. SC.Scan.NomNadir.* also use an index that must follow the same sequence of scan nadir class as used as a reference here.
Index, j, ranges from PD.LoIndx.RgAmb(2) to PD.HiIndx.RgAmb(2), that is, generally from 1 to the number of range ambiguity values in the scan(s) in the scan nadir class that has the most ambiguity numbers. This is the same range used for SC.RgAmb.Num.*Il*(isnc,j).

Units/representation
Float, radians

Treatment/usage
In order that the data model in this document not limit the ability of vendors to optimize performance by permitting a variety of ambiguity numbers to be used over the scans in a collection operation, we need to provide a way to indicate the ambiguity number for data received in a particular IPP. This ambiguity number is used to identify the IPP number of the transmitted pulse that is being received, and eventually to determine the range for returns in a given IPP's receive window. Rather than make a highly redundant[243] and very large list of ambiguity number values by IPP, we provide a map to the ambiguity number for an IPP based on the scan phase and scan nadir class associated with the IPP.

In the case that the ambiguity number changes with scan phase, we need to indicate which pulse ranges within the scan get which ambiguity number. We know that the first pulse of a scan will use the first ambiguity number, so we need only to specify the last pulse that uses the first ambiguity number to define the pulse range for the first ambiguity number. Subsequently, the last pulse of the next range is all that is necessary to specify the pulse range that uses the second ambiguity number. And so on…

For the **s**can **n**adir **c**lass indexed by "isnc",
SC.Scan.PhaseBins(isnc,j) = the maximum phase in the jth phase bin, to which a single ambiguity number value is assigned.

SC.Scan.PhaseBins(isnc,j) is used by the application that converts time of arrival data into round trip times, which might then be used to determine XYZ

[243] Even the repeat factor method used in the internal mapping of tiled data described in this document is insufficient to fully reduce the repeated ambiguity number data entries associated with cyclical variation of the ambiguity number within and across scans. We make double use of the mapping of IPPs to scans: it supports recovery of ambiguity numbers as well as documenting the instantatneous scene coverage rates for scanning systems and supporting the tracking of corrections due to internal registrations in building the point clouds.

positions for lidar returns. In order to determine the ambiguity number to be used for a particular IPP, one first must determine the scan nadir class for the scan containing that IPP by consulting SC.Scan.NadirClass.*ChNum*(is) and SC.Pulse.Scan.Num.*TileNum*(ip, j). One then consults SC.Pulse.Scan.Phase.*TileNum*(ip, j) to obtain the scan phase for the IPP. Then one determines which ambiguity number unique scan phase bin the IPP falls into by checking SC.Scan.PhaseBins(isnc,j). Finally, the scan nadir class index and the scan phase bin index are used in SC.RgAmb.Num.*Il*(isnc,j) to look up the IPP specific ambiguity number.

Note that while the range of index, j, in the SC.Scan.PhaseBins(isnc,j) array must be sized to accomodate the scan nadir class with the most ambiguity number values, there can be scan nadir classes in the data set with fewer ambiguity numbers. Any unused ambiguity number slots for a scan nadir class will occur (consecutively, if there are more than one) at the end of the list of "j" values and must be populated with a negative number to indicate that the phase bin slot is actually not applicable to that scan nadir class. Note also, that when there is only one scan nadir class and it has only one ambiguity number, there is no need to consult SC.Scan.NadirClass.*ChNum*(is), SC.Pulse.Scan.Num.*TileNum*(ip, j), SC.Pulse.Scan.Phase.*TileNum*(ip, j) and SC.Scan.PhaseBins(isnc,j) to determine what scan nadir class and scan phase bin to use to determine the ambiguity number. Population of SC.Scan.NadirClass.*ChNum*(is) and the rest of those items would no longer be essential to being able to determine a roundtrip time for a pulse. The range of "isnc" is one value, as is the range of "j". The only ambiguity number for each illuminator for the data set will be found in SC.RgAmb.Num.*Il*(1,1) [or (0,0) if the indexing is coded to be zero-based].

Governing quality metrics

See Also PD.HiIndx.RgAmb(2) SC.RgAmb.Num.*Il*(isnc,j) SC.Pulse.Scan.Phase.*TileNum*(ip, j)

20.5.77 SC.Scan.Type[L1,2,3,4,5]

Brief Definition SC.Scan.Type is a code for the scan trace shape description.

indices None

Units/representation Dimensionless, UInt8

Treatment/usage SC.Scan.Type carries one of the following codes for the nominal scan trace shape.
1=>linear cross track
2=>linear along track

	3=> linear oblique 4=>conical
Governing quality metrics	Correct code must be chosen
See Also	10.1.2.4 MD.Sensor.LidarMode(2)

20.5.78 SC.Scan.U_Cone(iscp, ich, k) [L1,2]

Brief Definition	SC.Scan.U_Cone(iscp, ich,k) gives the sensor frame orientation of the iscpth scan cone parameterization value for the scan cone axis direction.
indices	Index, iscp, ranges from PD.LoIndx.Scan(2) to PD.HiIndx.Scan(2), that is from 1 to the number scan cone parameterizations for a collection. Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. This reiterates the issue identified in 10.1.3.1 where the relationship of channel specific scan parameters and channel documentation was discussed. If a data set is to have scan parameters provided, and the collection scan parameterization was channel specific, that data set would need to provide channel definitions back to those which were collected. Index, k=1,2 for direction cosines measured from the sensor x and y axes respectively.
Units/representation	Dimensionless, float
Treatment/usage	SC.Scan.U_Cone(iscp, ich,k) gives the direction of the scan elliptical cone axis relative to the sensor frame for each choice of scan parameterization index, “iscp”, by channel, “ich”. Refer to the appendix figure 10.1.2-3. When the scan cone has minor axis equal to zero, the scan cone collapses to a plane relative to the sensor frame. This axis then points to the center of the scan in angle space relative to the sensor. We use direction cosines of angles α_{scan_cone} (measured from sensor frame x-axis) and β_{scan_cone} (measured from sensor frame y-axis) illustrated in the diagram to specify a unit vector (thus, U_cone) in the sensor frame that is along the axis of the elliptical cone. These values populate the k=1,2 elements respectively for each value of “ispc” and “ich”. Note that while it is possible that the scan cone index changes in lock step with the scan nadir class, it is not necessary and so the index symbols “isnc” and “ispc” are both used to support the general case, although common use

	might actually be that these two indices are equivalent.
Governing quality metrics	This item should have the same sequence of row index values as the data for the scan major axis plane normal direction. Locking these indices together forms a table with consistent data on each row. The uncertainty in the values of the direction cosines is the percentage of error in the data item.
See Also	SC.Scan.N_Major(iscp,ich,k)

SC.Subswath.*

SC.Subswath.* parameters describe sensor configuration items related to *collection* subswaths

20.5.79 SC.Subswath.TotNum [L1,2]

Brief Definition	SC.Subswath.TotNum gives the number of collection subswaths (native collected scan groups) employed by the collector to create the data set.
indices	none
Units/representation	Dimensionless, UInt16
Treatment/usage	Per appendix 10.1.4.3 a sensor collection operation may have naturally grouped sequences of scans into *collection* subswaths, which taken together represent the entire collection dedicated to a given product data set. The number of such *collection* subswaths per the product swath is given by SC.Subswath.TotNum[244]. In addition to collection subswaths, the processor may create *processing* subswaths as part of a parallel processing or quality improvement strategy. When that is done, there may be several subswaths created within a subswath processing level, of which there may be more than one. Those are processing architecture dependent labels for aggregated data groups in the data set, generally kept around only during processing. The *collection* subswaths are similar in that a collection subswath is a consecutive block of adjacent scans. The *collection* subswaths are different, however, in the sense that there are no arbitrary processing choices involved. There are no arbitrary levels of scan aggregation that contribute to identification of a *collection* subswath. The sensor collection strategy created a geometrical relationship between the scans in a single collection subswath and distinct from scans in other collection subswaths. A processor may further impose a processing subswath structure on the data as an interim data organization to facilitate assignment of work to distinct processing nodes, but it should honor the collection subswaths in its aggregation process so that natural boundaries in the data as collected may be preserved. Thus, it is recommended that a processing subswath boundary not fall within a collection subswath unless the

[244] This means that if a product at L2 or higher represents only part of the full collected swath, SC.Subswath.TotNum refers to only the number of collection subswaths represented in the data set.

	subswath level is low enough that multiple processing subswaths composite to complete collection subswaths.
Governing quality metrics	The number must be correct.
See Also	PF.Subswath.NumSubswaths(iswl)

20.5.80 SC.Subswath.NumScansPer [L1,2]

Brief Definition	SC.Subswath.NumScansPer gives the number of scans per collection subswath.
indices	none
Units/representation	Dimensionless, UInt32
Treatment/usage	This parameter is only populated when there are collection subswaths. When present, SC.Subswath.NumScansPer indicates that there are collection subswaths and how many scans there are in each. All collection subswaths must have the same number of scans, with the possible exception of the last collection subswath, which could be incomplete without causing a serious problem for processsing. A collection that utilizes collection subswaths will have the first SC.Subswath.NumScansPer scans constitute the first collection subswath. The next SC.Subswath.NumScansPer scans make up the second subswath and so on. The metadata structure supporting subswath documentation is simplified by the requirement that processing subswaths at the lowest level composite to the collection subswaths before higher levels of aggregation are created. This requirement is not significantly restrictive on the processor and represents a best practice that would naturally be used in processing of point cloud data from initial pulse by pulse angle and time-delay space to XYZ in a ground frame. As a consequence, SC.Subswath.NumScansPer has the same value as populates PF.Subswath.NumScans(iswl) for some low value of iswl, likely 1, 2, or 3. The index iswl counts the level of subswath aggregation. Thus a process that adopted the collection subswaths as the first level of subswath aggregation would have SC.Subswath.NumScansPer= PF.Subswath.NumScans(1). On the other hand, if the collection subswaths are quite large compared to H/W resources[245], the processor might begin with smaller subswaths and the equality condition could arise such that SC.Subswath.NumScansPer= PF.Subswath.NumScans(2). The use of a first level of subswath that is smaller than a collection subswath would be a good approach if scan to scan registration were needed even within a single collection subswath. The first processing subswath level might be set to the

[245] Or blocks of data with highly correlated errors

	single scan level of data and the second level of subswaths would match up to the collection subswaths.
Governing quality metrics	Values correctly populated
See Also	PF.Subswath.NumScans(iswl)

SC.Swath.*

SC.Swath.* parameters describe sensor configuration items related to the data set collection operation as a whole.

20.5.81 SC.Swath.FineTimeCal.*Ia*(ie,j) [L1,2]

Brief Definition	SC.Swath.FineTimeCal.*Ia*(ie,j) gives a correction that when added to the roundtrip time calculation for a return compensates for measured delays within the sensor electronics that bias the roundtrip calculations away from true time.
indices	Substring, "*Ia*", acts as an index and the character(s) represents integers in the range from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of focal plane "arrays"[246]. This is an evaluated substring so that the length of the lists of elements can be adjusted to the needs of each array. Index, ie, ranges from PD.LoIndx.Array.Elem(ia,1) to PD.HiIndx.Array.Elem(ia,1), that is over the elements in the array whose index is the integer conversion of the symbols carried in evaluated string, *Ia*. If PD.LoIndx.Array.Elem(ia,1) has the value, 1, then PD.HiIndx.Array.Elem(ia,1), the last element index of array, ia, equals the number of elements in that array. Index, j=1,2
Units/representation	seconds, float
Treatment/usage	The fast time timing model in 10.1.1.1 as elaborated for round trip time calculation in SC.Pulse.TxDelay.*TileNum*(ip) indicates that roundtrip time calculation incorporates fast time index translations, offsets and conversions. These are based on the integer fast time sample clock values, but there may be known delays that are smaller than a fast time sampling clock step that should be added to meet the best accuracy available with the sensor data and laboratory calibration data. This is the purpose of SC.Swath.FineTimeCal.*Ia*(ie,1). If the time associated with the photon arrival incorporates an information

[246] Quotes are a reminder that an array has N elements and that N may be equal to one in some cases. This is a generalized concept of array.

propagation delay within the sensor that can be corrected, do it with this data item. This provides fine calibration of roundtrip ranges that could be critical to getting the best results in some modes, for example, pulse compression. The value is always added, but could well be a negative number.
This data item would also support the function of documenting fast time fine scale alignment of responses from multiple channels (because the array number is part of the channel definition).

SC.Swath.FineTimeCal.*Ia*(ie,2) is the uncertainty in the value of SC.Swath.FineTimeCal.*Ia*(ie,1).

Governing quality metrics: Compliance to definition. Accuracy (true step size-populated step size)/true step size

20.5.82 SC.Swath.MeanPhtnFlx.Dat(j) [L1,2]

Brief Definition: SC.Swath.MeanPhtnFlx.Dat(j) indicates the average photon flux incident on the photo detector array and the control time constant.

indices: Index, j, =1,2

Units/representation: Units determined by SC.Swath.MeanPhtnFlx.Idat(1 and 3) , Float

Treatment/usage: To fully support characterization of the sensor response, it is necessary to indicate the mean number of photons per second incident on a detector element on the focal plane. SC.Swath.MeanPhtnFlx.Dat(1) carries a per average pixel, time averaged incident flux measure.

The optical bench may use attenuators and iris style F-stop control to adjust the mean flux at the focal plane in order to place the sensor in the desired operating range. These settings compensate for the steady impact during a collection operation of optical bench features such as collimators, beam splitters, waveplates etc. but which could be variable across collection operations due to collection mode sensor configuration activity. While the photon flux level can be an essential element of a sensor radiometric calibration process, it also affects the ranging process in a material way. For example, a Geiger mode sensor has a probability of firing at a range of times for a scattering surface at a fixed range from the sensor. If the time resolution of the focal plane output recording system is fine enough, the sensor's range signature for a constant range surface may be resolved and the response depends on the product of the incident flux and the probability of an average detector firing per incident photon.

The averaging performed to provide a value for SC.Swath.MeanPhtnFlx.Dat(1) is over the field of view and the properly operating focal plane elements and a period of time long enough to smooth out sensor jitter and stochastic sensitivities and photon arrival statistics, scene and propagation effects such as scintillation fading, range to the scene, and spatially variable reflectivity.

SC.Swath.MeanPhtnFlx.Dat(2) carries the integration time interval for computing the adjustments to hold the mean incident flux constant.

Governing quality metrics

See Also SC.Swath.MeanPhtnFlx.Dat

20.5.83 SC.Swath.MeanPhtnFlx.Idat(j) [L1,2]

Brief Definition SC.Swath.MeanPhtnFlx.Idat(j) describes the SC.Swath.MeanPhtnFlx.Dat

indices Index, j= 1,2

Units/representation Dimensionless, UInt8

Treatment/usage SC.Swath.MeanPhtnFlx.Idat(1) describes flux the metric.

SC.Swath.MeanPhtnFlx.Idat(1) =0 => photons incident per second on one average detector. This is well suited to Geiger mode or photon counting devices where precision is important and flux levels are low.

SC.Swath.MeanPhtnFlx.Idat(1) =1 => $\log_{10}$ of photons incident per second on one average detector. This is well suited to photocurrent measuring devices where the incident flux is so high that a single precision floating point number can not capture the magnitude of the flux in photons per second.

SC.Swath.MeanPhtnFlx.Idat(2) describes the gain control mechanism to assure the stability of the mean photon flux.

SC.Swath.MeanPhtnFlx.Idat(2) = 0 => attenuation is fixed based on a calibration interval preceding the data collection operation

SC.Swath.MeanPhtnFlx.Idat(2) = 1 => attenuation is fixed based on a database value for the target and operating range and weather

SC.Swath.MeanPhtnFlx.Idat(2) = 2 => attenuation follows the flux level with an open loop gain control which responds to arriving flux measurements made by a dedicated flux sensor independent of the lidar focal plane detectors.

SC.Swath.MeanPhtnFlx.Idat(2) = 3 => attenuation follows the flux level with a closed loop gain control which responds to the actual recorded sensor output levels.

SC.Swath.MeanPhtnFlx.Idat(3) describes the gain control temporal responsiveness to assure the stability of the mean photon flux.

SC.Swath.MeanPhtnFlx.Idat(3) = 0 => the integration time interval in SC.Swath.MeanPhtnFlx.Dat(2) is a number of seconds
SC.Swath.MeanPhtnFlx.Idat(3) = 1 => the integration time interval in SC.Swath.MeanPhtnFlx.Dat(2) is a number of IPPs

Governing quality metrics

See Also

20.5.84 SC.Swath.Pos.X_Fac [L1,2,3,4,5]

Brief Definition: SC.Swath.Pos.X_Fac is the coefficient of the x-axis data in the linear transformations that produce x positions in meters from stored x-components of the lidar sensor position.

indices: none

Units/representation: Meters (per data step), float

Treatment/usage: In order to allow minimization of the data volume, lidar sensor position data is stored such that it is restored to SI unit positions referenced to the ECEF frame origin via a linear transformation of the form:

Xmeters=Xdat * X_Fac +X_Offset

In the above transformation, SC.Swath.Pos.X_Fac provides the scale factor, X_Fac and Xdat is supplied via SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,1). While we permit the X_Offset to vary from tile to tile in order to support use of smaller wordsizes for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j), the x-scaling is global for the data set. This makes the relative distances in the pre-scaled data uniform and permits trivial metadata support for unit scaling on a global basis. If SC.Swath.Pos.X_Fac is not populated, it is assumed to be 1.0E-02 (That is, default to centimeter granularity for sensor position data.).

Governing quality metrics: The value must be the exact value used by the processor to generate SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j)

See Also: SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Pulse.Pos.X_Offset(j_tile)

20.5.85 SC.Swath.Pos.Y_Fac [L1,2,3,4,5]

Brief Definition: SC.Swath.Pos.Y_Fac is the coefficient of the y-axis data in the linear transformations that produce y positions in meters from stored y-components of lidar sensor position.

indices: none

Units/representation: Meters (per data step), float

Treatment/usage: In order to allow minimization of the data volume, lidar sensor position data is stored such that it is restored to SI unit positions referenced to the ECEF frame origin via a linear transformation of the form:

Ymeters=Ydat * Y_Fac +Y_Offset

In the above transformation, SC.Swath.Pos.Y_Fac provides the scale factor, Y_Fac and Ydat is supplied via SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,2).

	While we permit the Y_Offset to vary from tile to tile in order to support use of smaller wordsizes for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j), the y-scaling is global for the data set. This makes the relative distances in the pre-scaled data uniform and permits trivial metadata support for unit scaling on a global basis. If SC.Swath.Pos.Y_Fac is not populated, it is assumed to be 1.0E-02.
Governing quality metrics	The value must be the exact value used by the processor to generate SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j)
See Also	SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Pulse.Pos.Y_Offset(j_tile)

20.5.86 SC.Swath.Pos.Z_Fac [L1,2,3,4,5]

Brief Definition	SC.Swath.Pos.Z_Fac is the coefficient of the z-axis data in the linear transformations that produce z positions in meters from stored z-components of lidar sensor position.
indices	none
Units/representation	Meters (per data step), float
Treatment/usage	In order to allow minimization of the data volume, lidar sensor position data is stored such that it is restored to SI unit positions referenced to the ECEF frame origin via a linear transformation of the form: Zmeters=Zdat * Z_Fac +Z_Offset In the above transformation, SC.Swath.Pos.Z_Fac provides the scale factor, Z_Fac and Zdat is supplied via SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,3). While we permit the Z_Offset to vary from tile to tile in order to support use of smaller wordsizes for SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j), the z-scaling is global for the data set. This makes the relative distances in the pre-scaled data uniform and permits trivial metadata support for unit scaling on a global basis. If SC.Swath.Pos.Z_Fac is not populated, it is assumed to be 1.0E-02.
Governing quality metrics	The value must be the exact value used by the processor to generate SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j)
See Also	SC.Pulse.SensorPos.*ChNum*.*TileNum*(ip,j) SC.Pulse.Pos.Z_Offset(j_tile)

20.5.87 SC.Swath.RateFactor(ia) [L1,2]

Brief Definition	SC.Swath.RateFactor(ia) gives a factor that when divided into the command clock step gives the data capture clock step.
indices	Index, ia, runs from PD.LoIndx.Array(1) to PD.HiIndx.Array(1), that is, nominally from 1 to the number of arrays described in the metadata for the current data set.
Units/representation	Dimensionless. UInt32
Treatment/usage	Per the time model discussion of Appendix 10.1.1, SC.Swath.RateFactor(ia)

is an integer factor defined as the ratio of the number of data capture clock steps per second to the number of sensor command clock steps per second. That is, the data capture clock runs faster than the command issue clock by a factor of SC.Swath.RateFactor which is greater than one. This value may be supplied for each array in the sensor.

The command clock step, CCStep, is provided in SC.Swath.TimeStep.
The rate factor , abbreviated here as "Rfac", is given for each sensing array by SC.Swath.RateFactor(ia).

The data capture clock step, DCStep, for any given array, ia, equals CCStep/Rfac.

Governing quality metrics: Proper value supplied

See Also: SC.Swath.TimeStep PD.HiIndx.Array(j)

20.5.88 SC.Swath.RecordDelay[L1,2]

Brief Definition: SC.Swath.RecordDelay is the number of fast data capture clock cycles after a command is issued to record data within an IPP that recording actually commences.

indices: none

Units/representation: Dimensionless, UInt16

Treatment/usage: Per the timing model in 10.1.1, the determination of photon round trip time of flight is dependendent on command times and delays from those command times. Command times come precisely on command clock ticks and since these are long intervals compared to light travel times over distances that are contained within the sensor device, more precise delays that might be represented in units of seconds or the faster data capture clock steps need to be provided to permit precise time calculations. The relation of SC.Swath.RecordDelay to other events in the IPP is shown in figure 10.1.1-2 and in the figure contained in SC.Pulse.TxDelay.*TileNum*(ip).
Note that this is in the highest precision measured by the sensor, the fast time data capture clock steps. However, this value may be effectively corrected by laboratory calibration measurements that may exceed the sensor's measurement precision. Thus, the discrete values of SC.Swath.RecordDelay the discrete might be subject to a further refinement by the addition of a correction in seconds. That correction, when available, is provided at the array element level by SC.Swath.FineTimeCal.*Ia*(ie).

Note that this item is important for the linear mode family where waveforms are sampled to record return flux levels as a function of time. It determines

	the fast data capture clock interval when recording commences and sets a time origin for roundtrip time construction/record keeping. It performs the same function in the Geiger Mode case. In GM, the recorded value is already a clock count referenced to the fast time capture interval specified by SC.Swath.RecordDelay.
Governing quality metrics	Proper value populated
See Also	SC.Pulse.TxDelay.*TileNum*(ip). SC.Swath.FineTimeCal.*Ia*(ie)

20.5.89 SC.Swath.SigmaClkStep [L1,2]

Brief Definition	SC.Swath.SigmaClkStep is the uncertainty in seconds for a sensor command clock step interval.
indices	none
Units/representation	Seconds, float
Treatment/usage	The primary objective of populating SC.Swath.SigmaClkStep is to support determining the amount of range dependent range error that is introduced by errors in the clock rate. The ranging is performed by using a clock that runs SC.Swath.RateFactor(ia) times faster than the sensor command clock for each sensing array, indexed by "ia". Thus, the ranging clock for array "ia" has time steps that are SC.Swath.TimeStep/SC.Swath.RateFactor(ia). The rate factor is a fixed integer with no uncertainty, so the uncertainty in the fast time clock step is SC.Swath.SigmaClkStep/ SC.Swath.RateFactor(ia). Furthermore, clocks are subject to jitter, causing each time tick to have a variation from the mean time step (calculated over some large number of time steps). For a clock that has no drift, there is no significant linear trend in the individual jitter errors at each tick. Thus SC.Swath.SigmaClkStep is not associated with jitter at all. It represents what might be considered the timing calibration error. Clocks used for lidars will not have significant drift over a durations as short as a collection operation, so the mean jitter error is always close to zero over many pulses. However, the mean clock time step for a collection operation may be different than the value in SC.Swath.TimeStep. The standard deviation of the distribution of errors in SC.Swath.TimeStep over all operating conditions is provided by SC.Swath.SigmaClkStep. Consequently, the clock rate uncertainty is determined by SC.Swath.SigmaClkStep and the error in the measurement of round trip times grows with the range proportionally to SC.Swath.SigmaClkStep. The .95P value of the clock uncertainty is provided by the product of SC.Swath.SigmaClkStep and ES.Swath.P95GM_Fac(2).

For a particular round trip duration (τ_{rt}), the uncertainty in the round trip time is $\left\langle (\Delta\tau_{rt})^2 \right\rangle = \frac{\left\langle (\Delta\delta t_{step})^2 \right\rangle}{\delta t_{step}^{\ 2}} \tau_{rt}^{\ 2} + (\delta t_{step})^2 \left(\left\langle (\Delta n_{step2})^2 \right\rangle + \left\langle (\Delta n_{step1})^2 \right\rangle \right)$

Where $\Delta\delta t_{step}$ is SC.Swath.SigmaClkStep/SC.Swath.RateFactor(ia), δt_{step} is the sampling clock step obtained from SC.Swath.TimeStep and SC.Swath.RateFactor(ia), Δn_{step2} is the uncertainty in the fast time sample index for the received photon detection time and Δn_{step1} is the the uncertainty in the fast time sample index for the received photon transmission time. These latter values are integers, representing time indices, rather than seconds, for transmit and receive events. When multiplied by δt_{step}, the event time indices are converted to seconds.

The range dependent range uncertainty component due to the clock step error for the jth snapshot can be obtained by multiplying [SC.Swath.SigmaClkStep/ SC.Swath.RateFactor(ia)] by [ES.Scan.GM_SigmaR(j)/ES.Swath.SigmaT].

Governing quality metrics: SC.Swath.SigmaClkStep acts as a quality measure for SC.Swath.TimeStep. In this generation of the Lidar CMMD we do not offer the "error in the error" for "fast time" time keeping.

See Also: SC.Swath.RateFactor(ia) SC.Swath.TimeStep ES.Swath.P95GM_Fac(1) ES.Scan.GM_SigmaR(j) ES.Swath.GM_SigmaT

20.5.90 SC.Swath.TimeStep [L1,2]

Brief Definition: SC.Swath.TimeStep is the time in seconds for a sensor command clock step interval.

indices: none

Units/representation: Seconds, double float

Treatment/usage: Typically, ranging sensors use a clock to determine when to kick off IPP events such pulse transmission and data recording. The time interval between successive "ticks" of this clock is called the command clock step. This is often the master clock and data capture clocks are run as multiples of the master clock frequency.

In order to properly scale the dynamics of scenes measured via a lidar, more than just the pulse indexing of parameters such as the sensor position is necessary. Also, since the data capture step interval is given by a factor which scales it relative to the command clock step, the fast capture time intervals cannot be converted to time without knowing the SC.Swath.TimeStep. This conversion permits ranging in meters.

Governing quality metrics: (Actual time step- SC.Swath.TimeStep)/Actual time step

See Also: SC.Swath.RateFactor(ia)

20.5.91 SC.Swath.TimeSync[L1,2]

Brief Definition	SC.Swath.TimeSync is the time uncertainty in seconds for start time coordination with UTC or TJD referenced time lines.
indices	none
Units/representation	Seconds, float
Treatment/usage	The sensor system clock must be coordinated with external time references so that navigation and other types of data can be properly compared across systems. A start time for collection of data is available in a *precision* of at least 0.01 seconds [see MD.Collect.StartTime, MD.Aggregate.Parent.StartTime(j), MD.Aggregate.Fellow.StartTime(j)]. SC.Swath.TimeSync indicates the accuracy of the alignment of this time with UTC and the truncated Julian day midnight time by providing the 90th percentile error bar of the sensor system start time with respect to the universal time reference. Thus, the Julian time or UTC time corresponding to a given sensor time can be determined from the sensor based time to an accuracy of SC.Swath.TimeSync.
Governing quality metrics	
See Also	MD.Collect.StartTime, MD.Aggregate.Parent.StartTime(j), MD.Aggregate.Fellow.StartTime(j)]

SC.WF.*

SC.WF.* parameters describe sensor configuration items related to the Waveform capture.

20.5.92 SC.WF.RayDesc.ArryGlbl(iray) [L1,2]

Brief Definition	SC.WF.RayDesc.ArryGlbl(iray) associates an array choice with each global ray number. This is part of a ray definition table that pairs arrays and sensor look directions within the instantaneous field of view.
indices	Index, iray, ranges from PD.LoIndx.Array(8) to PD.HiIndx.Array(8), that is generally from 1 to the number of rays in the global array table.
Units/representation	Dimensionless, UInt32
Treatment/usage	SC.WF.RayDesc.ArryGlbl(iray), together with SC.WF.RayDesc.ElmtGlbl(iray), makes an assignment of the combination of a choice of a particular array and an element on that array to a single ray index, "iray". This assignment is **global** in the sense that a value of iray is unique no matter what tile it appears in when it is defined via SC.WF.RayDesc.ArryGlbl(iray). SC.WF.RayDesc.ArryGlbl(iray) must be populated when three conditions hold: 1. Waveform data is present in the data set

2. More than one array is represented in the data set
3. The rays (one ray per array and array element choice) are defined globally for all arrays and all elements in the arrays.

There are alternative situations that would result in leaving SC.WF.RayDesc.ArryGlbl(iray) unpopulated: where there are either no waveforms, or where the rays used are few enough that a strategy of only defining the ray index for utilized rays makes sense, or where there is only one channel in the data set.
When the conditions 1 and 3 above are not true, SC.WF.RayDesc.ArryGlbl(iray) would be left unpopulated. If only condition 3 is not true, as many instantiations of SC.WF.RayDesc.Arry.*TileNum*(iray) as necessary would be used instead.
When neither SC.WF.RayDesc.ArryGlbl(iray) nor SC.WF.RayDesc.Arry.*TileNum*(iray) are populated, all rays are assumed to belong to the only array reported.

The received range data vectors are tiled with indices (ipi,iray,it)[247]. The range of iray used in the 3-D range vector array must match that of the iray definition tables. So, defining all of the iray directions possible but using only a few is space wasting. The ray identification is needed for each range vector in a tile. A natural match for that need is to have a tile based array& element to iray table for each tile when only a few ray directions are used across all of the tiles, which is a possibility for Enterprise level-2 data sets. Only the pulse-element pairs that exist would get iray index definitions via the tables SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). On the other hand, when it can be expected that most tiles actually exhibit rays for nearly all of the rays (typical of enterprise level-1), a single global table is more efficient and we use SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) instead.

Governing quality metrics: Consistent labeling of rays with the recorded range data.

See Also: SC.WF.RayDesc.Arry.*TileNum*(iray) PD.HiIndx.Array(j) SC.Array.RayDirs.*Ia*(icase,ie,j) ES.WFClp.ChNum.*TileNum*(ipi,iray) LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it)

20.5.93 SC.WF.RayDesc.ElmtGlbl(iray) [L1,2]

Brief Definition: SC.WF.RayDesc.ElmtGlbl(iray) associates a global array element number with each ray representing a sensor look direction within the instantaneous

[247] There is no channel index because ES.WFClp.ChNum.*TileNum*(ipi,iray) provides channel identification for waveform data.

field of view.

indices
Index, iray, ranges from PD.LoIndx.Array(8) to PD.HiIndx.Array(8), that is generally from 1 to the number of rays in the global array table.

Units/representation
Dimensionless, UInt32

Treatment/usage
The *value* of SC.WF.RayDesc.ElmtGlbl(iray) is a single number that is the element index, "ie", of the physical focal plane detector array element by which the data in LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) was measured. As long as the data writing application keeps values of "ie" consistently assigned [see SC.Array.RayDirs.*Ia*(icase,ie,j)], all is well. The choice of "ie" correspondence to a specific physical element is arbitrary as long as it is unique, "ie" is a consecutive set of integers, and must be consistently used within a data set.

Thus, SC.WF.RayDesc.ElmtGlbl(iray), when populated, gives the detector array element index corresponding to the ray attached to the index value, iray, [conceptually, ray(iray)] for all references within the data set.

SC.WF.RayDesc.ArryGlbl(iray), together with
SC.WF.RayDesc.ElmtGlbl(iray),
makes an assignment of the combination of a choice of a particular array and an element on that array to a single ray index, "iray". This assignment is global in the sense that a value of iray is unique no matter what tile it appears in when it is defined via SC.WF.RayDesc.ElmtGlbl(iray).
SC.WF.RayDesc.ElmtGlbl(iray) must be populated under two conditions:
1. Waveform data is present in the data set
2. The rays (one ray per array and array element choice) are defined globally for all arrays and all elements arrays.

There are alternative situations where there are either no waveforms or where the rays used are few enough that a strategy of only defining the ray index for utilized rays makes sense.
When condition 2 above is not true, SC.WF.RayDesc.ElmtGlbl(iray) would be left unpopulated and as many instantiations of
SC.WF.RayDesc.Elmt.*TileNum*(iray) as necessary would be used instead.

The received range data vectors are tiled with indices (ipi,iray,it). The range of iray used in the 3-D range vector array must match that of the iray definition tables. So defining all of the iray directions possible but using only a few is space wasting. The ray identification is needed for each range vector in a tile. A natural match for that need is to have a tile based array& element to iray table for each tile when only a few ray directions are used across all of the tiles, which is a possibility for Enterprise level-2 data sets. Only the

pulse-element pairs that exist would get iray index definitions via the tables SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). On the other hand, when it can be expected that most tiles actually exhibit rays for nearly all of the rays (typical of enterprise level-1), a single global table is more efficient and we use SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) instead.

Governing quality metrics: Consistent labeling of rays with the recorded range data.

See Also: SC.WF.RayDesc.Elmt.*TileNum*(iray) PD.HiIndx.Array(j) LD.WFClp.RxDatVec.*TileNum*(ip,iray,it)

20.5.94 SC.WF.RayDesc.Arry.*TileNum*(iray) [L1,2]

Brief Definition: SC.WF.RayDesc.Arry.*TileNum*(iray) associates a tile specific ray number with each array choice paired with a particular sensor look direction within the instantaneous field of view.

indices: Index, iray, ranges from PD.LoIndx.Ray.*TileNum* to PD.HiIndx.Ray.*TileNum*, that is, from 1 to the number of rays in the tile indicated by *TileNum*.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Dimensionless, UInt16

Treatment/usage: SC.WF.RayDesc.Arry.*TileNum*(iray), together with SC.WF.RayDesc.Elmt.*TileNum*(iray), makes an assignment of the combination of a choice of a particular array and an element on that array to a single ray index, “iray”. This assignment is *tile specific* in the sense that a value of iray is uniquely mapped to an array and an element only within the tile identified by *TileNum*. Thus, when only a few rays have populated waveform clips, relatively low values of ‘iray” may be reused across tiles, with local meaning provided via SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). SC.WF.RayDesc.Arry.*TileNum*(iray) must be populated under three conditions:

1. Waveform data is present in the data set
2. More than one array is represented in the data set
3. The rays (one ray per array and array element choice) are defined specifically for the tiles in which they appear.

There are alternative situations where there are either no waveforms or where so many rays are used that a strategy of defining and using all of the possible ray indices makes sense, or where there is only one array in the data set.

When the conditions 1 and 3 above are not true, SC.WF.RayDesc.Arry.*TileNum*(iray) would be left unpopulated. When only condition 3 is untrue, SC.WF.RayDesc.ArryGlbl(iray) would be used instead. When neither SC.WF.RayDesc.ArryGlbl(iray) nor SC.WF.RayDesc.Arry.*TileNum*(iray) are populated, all rays are assumed to belong to the only array reported.

The received range data vectors are tiled with indices (ipi,iray,it). The range of iray used in the 3-D range vector array must match that of the iray definition tables. So defining all of the iray directions possible but using only a few is space wasting. The ray identification is needed for each range vector in a tile. A natural match for that need is to have a tile based array& element to iray table for each tile when only a few ray directions are used across all of the tiles, which is a possibility for Enterprise level-2 data sets. Only the pulse-element pairs that exist would get iray index definitions via the tables SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). On the other hand, when it can be expected that most tiles actually exhibit rays for nearly all of the rays (typical of enterprise level-1), a single global table is more efficient and we use SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) instead.

Governing quality metrics: Consistent labeling of rays with the recorded range data.

See Also: SC.WF.RayDesc.Arry.*TileNum*(iray) PD.HiIndx.Ray.*TileNum* SC.WF.RayDesc.ArryGlbl(iray)

20.5.95 SC.WF.RayDesc.Elmt.*TileNum*(iray) [L1,2]

Brief Definition: SC.WF.RayDesc.Elmt.*TileNum*(iray) associates a tile specific array element number with with each ray representing a sensor look direction within the instantaneous field of view that has waveform data included in the tile.

indices: Index, iray, ranges from PD.LoIndx.Ray.*TileNum* to PD.HiIndx.Ray.*TileNum*, that is, from 1 to the number of rays in the tile indicated by *TileNum*.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. This is as if the row in which a data item is found in a 2-D matrix were built into the name and only the column was given as an array index.

Units/representation: Dimensionless, UInt32

Treatment/usage: The *value* of SC.WF.RayDesc.Elmt.*TileNum*(iray) is a single number that is the element index, “ie”, of the physical focal plane detector array element by

which the data in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) was measured. As long as the data writing application keeps values of "ie" consistently assigned [see SC.Array.RayDirs.*Ia*(icase,ie,j)], all is well. The choice of "ie" correspondence to a specific physical element is arbitrary as long as it is unique, "ie" is a consecutive set of integers, and consistently used within a data set.

Thus, SC.WF.RayDesc.Elmt.*TileNum*(iray) gives the detector array element index corresponding to ray(iray) for all references within a specific tile, indicated by *TileNum*.

SC.WF.RayDesc.Arry.*TileNum*(iray), together with SC.WF.RayDesc.Elmt.*TileNum*(iray), makes an assignment of the combination of a choice of a particular array and an element on that array to a single ray index, "iray". This assignment is tile specific in the sense that a value of iray is uniquely mapped to an array and an element only within the tile identified by *TileNum*. Thus, when only a few rays have populated waveform clips, relatively low values of 'iray" may be reused across tiles, with local meaning provided via SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray).
SC.WF.RayDesc.Elmt.*TileNum*(iray) must be populated under two conditions:
1. Waveform data is present in the data set
2. The rays (one ray per array and array element choice) are defined specifically for the tiles in which they appear.

There are alternative situations where there are either no waveforms or where so many rays are used that a strategy of defining and using all of the possible ray indices makes sense. When condition 2 above is not true, SC.WF.RayDesc.Elmt.*TileNum*(iray) would be left unpopulated SC.WF.RayDesc.ElmtGlbl(iray) would be used instead.
When neither SC.WF.RayDesc.ElmtGlbl(iray) nor SC.WF.RayDesc.Elmt.*TileNum*(iray) are populated, the data set will not have waveform clips.
The received range data vectors are tiled into arrays with indices (ipi, iray, it). The range of iray used in the 3-D range vector array must match that of the iray definition tables. So defining all of the iray directions possible but using only a few is wasteful. The ray identification is needed for each range vector in a tile. A natural match for that need is to have a tile based array& element to iray table for each tile when only a few ray directions are used across all of the tiles, which is a possibility for Enterprise level-2 data sets. Only the pulse-element pairs that exist in a specific tile would get iray index definitions

	via the tables SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). On the other hand, when it can be expected that most tiles actually exhibit rays for nearly all of the rays (typical of enterprise level-1), a single global table is more efficient and we use SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) instead.
Governing quality metrics	Consistent labeling of rays with the recorded range data.
See Also	SC.WF.RayDesc.ElmtGlbl(iray) SC.Ray.HiIndx

20.5.96 SC.WF.Rx.Nsamps(ich) [L1,2]

Brief Definition	SC.WF.Rx.Nsamps(ich) indicates the length of the each channel's receive window in terms of fast time data capture clock intervals.
indices	Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set.
Units/representation	Dimensionless, UInt32
Treatment/usage	SC.WF.Rx.Nsamps(ich) indicates the (possibly channel specific) length of the receive window in terms of fast time data capture clock intervals. This meaning is the same for both linear and Geiger mode operation. The significance is different, however. For linear mode, the number of recorded samples in a full waveform capture would be the same as the number of fast clock intervals in the receive window, as given by SC.WF.Rx.Nsamps(ich). In Geiger mode, there is only one recorded photon arrival time recorded per quench. But the number[248] of times to which the photon arrival could be quantized is given by SC.WF.Rx.Nsamps(ich). Note that if a quench's outage occurs during the receive window, the detector will not be responsive to photons and some of the SC.WF.Rx.Nsamps(ich) fast clock intervals in the receive window can't actually be output arrival times. In GM, this is still the duration of the receive window in fast clock steps.
Governing quality metrics	(True # of samps-reported number of samps)/ True # of samps --objective value should be 0.
See Also	unification

20.5.97 SC.WF.Rx.IntnsyUncty(ich) [L1,2,3,4,5]

Brief Definition	SC.WF.Rx.IntnsyUncty(ich) gives the standard deviation of the measured relative intensity due to sensor error after deterministic corrections have been applied.

[248] For a multi quench system, there could be some blind times. So in such a case, strictly, not all sample times can be output times, as explained in the following sentences.

indices	Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set.
Units/representation	Dimensionless, Float
Treatment/usage	SC.WF.Rx.IntnsyUncty(ich) gives the channel specific residual uncertainty in relative intensity due to sensor error after application of SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) and excluding scene and propagation effects. This figure is the standard deviation of intensity measurement averaged over the array associated with the channel. This uncertainty in intensity of sensor origin after application of SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k) represents error bars around any point's relative intensity derived from the lidar data.
Governing quality metrics	
See Also	SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)

20.5.98 SC.WF.TxProfile.FM_Slope(il) [L1,2]

Brief Definition	SC.WF.TxProfile.FM_Slope(il) is the slope of the frequency ramp when linear frequency modulation is used to permit lidar pulse compression.
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of illuminating lasers used in distinguishing channels.
Units/representation	s^{-2}, Float Note that typical radio spectrum frequency slopes to support range resolutions of interest for pulse compression of lidar will be in the MHz/microsec (=THz/sec) range so that typical values of SC.WF.TxProfile.FM_Slope(il) would be in the 10^{12} s^{-2} range.
Treatment/usage	Geiger mode is a design intended to avoid requiring very high power to achieve short range resolution intervals. An alternative is pulse compression, which is a standard approach for radar. In this method, the transmit pulse would be given a modulation over a long duration compared to the light travel time across the minimum desired range resolved swath interval. Linear frequency modulation of an RF amplitude modulation envelope on the optical carrier frequency is an established approach. So is[249] modulation of the optical carrier frequency. SC.WF.TxProfile.FM_Slope is a key parameter of such a modulation. The range width of a compressed pulse depends on the bandwidth only (in the

[249] Polarization could be used too. However we treat here frequency modulation that is to be supported by the linear FM metadata group, of which this is an element.

	limit that the pulse is long enough that the carrier of the modulation has many oscillations within the pulse). Thus, we expect lidar usage with modulation bandwidths in the neighborhood of GHz. These are accessible with very short pulses if the optical frequency is modulated, rather than an imposed RF amplitude modulation of the optical signal. The lidar choice for modulation of the optical frequency versus modulation of an RF amplitude modulation of the optical carrier wave is conveyed in SC.WF.TxProfile.Flags(2). The frequency modulation rate provided by SC.WF.TxProfile.FM_Slope(il) applies in either case. The transmitted bandwidth is then the product of the linear FM rate in SC.WF.TxProfile.FM_Slope(il) and the transmitted pulse duration. The transmitted pulse duration is to be determined from the product of SC.WF.TxProfile.Nsamps(il) and the sampling clock step, which is discussed in SC.Swath.RateFactor(ia)[250].
Governing quality metrics	(True frequency slope-reported frequency slope)/(True frequency slope). The objective value should be 0.
See Also	PD.HiIndx.Laser(1) SC.WF.TxProfile.Flags(2) SC.Swath.RateFactor(ia)

20.5.99 SC.WF.TxProfile.FM_Start(il) [L1,2]

Brief Definition	SC.WF.TxProfile.FM_Start(il) is the starting frequency of the FM ramp when linear frequency modulation is used to permit lidar pulse compression.
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of illuminating lasers used in distinguishing channels.
Units/representation	Hz, Float
Treatment/usage	An alternative to Geiger mode as a mitigation against requiring very high power to achieve short range resolution intervals is pulse compression, which is a standard approach for radar. In this method the transmit pulse would be modulated over a long duration compared to the light travel time across the minimum desired range resolved swath interval. Both linear frequency modulation of an RF amplitude modulation envelope on the optical carrier beam and actual modulation of the optical frequency itself are established frequency modulation approaches. SC.WF.TxProfile.FM_Start is a key parameter of FM modulation with either modulation technique. When the modulation center is the photonic frequency, we will see values of SC.WF.TxProfile.FM_Start(il) of the order of 1.0E+14 Hz (hundreds of THz), while a ramp imposed on a carrier modulation in the MHz region would start in the neighborhood of 1.0E+06 Hz.
Governing quality	(True start frequency -reported startfrequency)/ True start frequency --

[250] These items should be populated to support pulse compression mode even when the transmitted pulse is not presented in SC.WF.TxProfile.Dat(il,j).

metrics	objective value should be 0.
See Also	SC.WF.TxProfile.FM_Slope

20.5.100 SC.WF.TxProfile.FM_PhaseStart(il) [L1,2]

Brief Definition	SC.WF.TxProfile.FM_PhaseStart(il) is the starting phase when linear frequency modulation is used to permit lidar pulse compression.
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of illuminating lasers used in distinguishing channels.
Units/representation	degrees, Float
Treatment/usage	An alternative to Geiger mode as a mitigation against requiring very high power to achieve short range resolution intervals is pulse compression, which is a standard approach for radar. In this method the transmit pulse would be modulated over a long duration compared to the light travel time across the minimum desired range resolved swath interval. Both linear frequency modulation of an RF amplitude modulation envelope on the optical carrier beam and actual modulation of the optical frequency itself are established frequency modulation approaches. SC.WF.TxProfile.FM_PhaseStart is a key parameter of such a modulation. SC.WF.TxProfile.FM_PhaseStart gives the phase of the wave form modulation at the start of the transmitted frequency ramp.
Governing quality metrics	(True start phase - reported start phase)/(True start phase) --objective value should be 0.
See Also	SC.WF.TxProfile.FM_Slope

20.5.101 SC.WF.TxProfile.Dat(il,j) [L1,2]

Brief Definition	SC.WF.TxProfile.Dat(il,j) is the (fast) time sampled transmitted lidar pulse wave form. The captured form of the transmit pulse depends on the lidar time encoding mode.
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of lasers. Index, j, runs from from 1 to SC.WF.TxProfile.Nsamps(il)
Units/representation	Dimensionless, UInt(variable given by PD.Wordsize.TxDat)
Treatment/usage	The transmitted wave form is likely to be captured in one of two forms. For linear mode and Geiger mode operation where no modulation is intentionally imposed, fast time samples of the intensity of the transmitted pulse are recorded. If a time modulation to support pulse compression has been imposed, a representation of the transmitted wave form is convenient for the case where simple parameters cannot adequately describe the transmitted wave form. Examples of cases where simple parameters suffice are linear

FM and polarization modulation.

For linear or Geiger mode, thorough processing and proper interpretation of point cloud artifacts requires the transmitted radiant intensity vs time to be reported with a dynamic range and time sample density good enough to define shapes of multiple intensity peaks if they exist.
The fast time sampling time step will be given by SC.Laser.Dat(il,9). It is recommended that this be no greater than the step due to the the Rx sample rate determined from the rate factor, SC.Swath.RateFactor(ia). If not all of the arrays sample laser "il", at the same rate, SC.WF.TxProfile.Dat(il,j) will be populated at the highest sampling rate among the arrays that record laser "il".
For recorded intensity, SC.WF.TxProfile.Dat(j) must be multiplied by SC.WF.TxProfile.Peak to obtain physical units[251]. The basis of normalization of SC.WF.TxProfile.Dat(j) is provided in SC.WF.TxProfile.Flags(1), but no account is made for pulse to pulse overall transmitted power fluctuation in SC.WF.TxProfile.*. That function may[252] be incorporated into pulse level corrections in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) which multiplies LD.WFClp.RxDatVec.TileNum(ip,iray,it) to apply deterministic sensor response corrections on a pulse-ray basis. This method combines the transmit power and pulse dependent sensitivity with an ambiguity number shift correction and is convenient when there is a low likelihood that the documented ambiguity number is incorrect or that the error in the ambiguity number would make an error budget breaking difference. See discussion in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1).

For the case of pulse compression lidar the fast time sampling rate will again be the same as for the received waveforms, but the samples would be VQ encoded samples of the demodulated waveform. The choice of VQ table to encode the transmitted pulse data shall be the one with the largest VQ wordsize among the channels that laser "il" participates in.

Governing quality metrics

See Also

SC.WF.TxProfile.Nsamps(il) PD.HiIndx.Laser(1) PD.Wordsize.TxDat SC.Laser.Dat(il,9) SC.WF.TxProfile.Peak SC.Laser.IDat(il,5)

[251] Conversion to physical units is not needed for non-radiometric exploitation.

[252] This would be the case when SC.Laser.IDat(il,5) = 1 and SC.WF.TxProfile.Flags(4)=0. Another case is indicated by SC.Laser.IDat(il,5) = 1 and SC.WF.TxProfile.Flags(4)=1. In that case, a metadata item deferred in this version of the CMMD, SC.Pulse.TxPeakTracker(il,ip), would provide pulse to pulse power level corrections that are independent of the element sensivity data and have no ambiguity number shift built in. Whenever SC.Laser.IDat(il,5) = 0, transmit power corrections for laser "il" are not documented in the data set in any manner.

SC.WF.TxProfile.Flags(j) SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1)
SC.WF.VQ_Decode(ich,icode)

20.5.102 SC.WF.TxProfile.Flags(j) [L1,2]

Brief Definition: SC.WF.TxProfile.Flags(j) lists a number of codes that describe the transmitted pulse

indices: j ranges from PD.LoIndx.Array (9) to PD.HiIndx.Array (9), that is generally from 1 to the number of flags defined (current value=4)

Units/representation

Treatment/usage: j=1 Profile envelope code

=1 → profile is a long term average of many pulses
=2 → profile is worst case(1) envelope of many pulses (max intensity over ensemble at each sample time into the pulse)
=3 → profile is worst case(2) envelope of many pulses (min intensity over ensemble at each sample time into the pulse)
=4 → profile is worst case(3) envelope of many pulses (at each sample time into the pulse, present max over ensemble of points below the pulse intensity average and min over ensemble of points above the pulse intensity average) minimizes high output intervals and maximizes low output intervals.

j=2 Modulation type

=0 → None imposed
=1 → RF amplitude on optical carrier (That is, the photons all have the same wavelength.)
=2 → RF frequency modulation of optical carrier (That is, the photon wavelength varies through the pulse.)
=3 → Polarization (linear) ramp 45 degrees total modulation
=4 → Polarization (linear) ramp 60 degrees total modulation
=5 → Polarization (linear) ramp 90 degrees total modulation

j=3 Squeezed parameter

=0 → None imposed
=1 → amplitude
=2 → phase
=3 → vacuum

j=4 Indicates where transmit power fluctuations are tracked, when tracking is indicated by SC.Laser.IDat(il,5)

= 0→ Transmit pulse power tracking is incorporated with ambiguity number shift in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)
=1→ Transmit pulse power tracking would be in deferred item, SC.Pulse.TxPeakTracker(il,ip), which is not ambiguity

	number shifted
Governing quality metrics	
See Also	PD.HiIndx.Array (9)

20.5.103 SC.WF.TxProfile.Nsamps(il) [L1,2]

Brief Definition	SC.WF.TxProfile.Nsamps(il) gives the length of the Transmit profile data vector
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of illuminating lasers used in distinguishing channels.
Units/representation	Dimensionless, UInt32
Treatment/usage	number of samples in the fast time Tx pulse trace
Governing quality metrics	Right value or not
See Also	SC.WF.TxProfile.Dat(j)

20.5.104 SC.WF.TxProfile.Peak(il) [L1,2]

Brief Definition	SC.WF.TxProfile.Peak(il) is the conversion factor to remove scaling of SC.WF.TxProfile.Dat(j).
indices	Index, il, ranges from PD.LoIndx.Laser(1) to PD.HiIndx.Laser(1), that is, nominally from 1 to the number of illuminating lasers used in distinguishing channels.
Units/representation	Watts/sample unit, float
Treatment/usage	SC.WF.TxProfile.Dat(j) is the factor that multiplies SC.WF.TxProfile.Dat(j) to yield physical units. SC.WF.TxProfile.Dat(j) will be proportional to photo current output of a measuring devices that sees the full output beam, which in turn is proportional to transmitted power. For analysis that depends only on relative beam power levels, this value is not necessary. Subtleties regarding the limitations of the calibration based on using SC.WF.TxProfile.Dat(j) are discussed in SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1) and SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1).
Governing quality metrics	
See Also	SC.WF.TxProfile.Dat(j) SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,1)

20.5.105 SC.WF.VQ_Decode(ich, icode) [L1,2]

Brief Definition	SC.WF.VQ_Decode(ich,icode) provides look up decoding of vector quantized waveform data generated by pulse compression lidar on a channel basis.

indices	Index, ich, ranges from PD.LoIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.LoIndx.Channlzn.Rg_m), that is, over the channels of the original raw data set. Index, icode, ranges from PD.LoIndx.Array(11) to PD.HiIndx.Array(11), that is, over the number of VQ decode table entries in the longest VQ decode table among the channels. In the event that there are channels with different length VQ decode tables, the trailing entries (values of icode beyond the end of meaningful data) will be set to 0. Proper software coding practice should result in these elements never being referenced.
Units/representation	Dimensionless, complex single precision float
Treatment/usage	VQ decoding is indicated as necessary when MD.Sensor.lidarMode(1) is set to 3. The use of the table, SC.WF.VQ_Decode(ich,icode), is as a look up translation. The recorded data item in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) or SC.WF.TxProfile.Dat(il,j) must be placed in an integer form if it is not already 8 or 16 bits in length. The recorded bit string is to be interpreted as the least significant bits of the integer. Use the converted value in the range vector as a look up in this table. For channel "ich", the complex number output for an encoded sample in the "Dat"[253] vector at a fast time index value 'it" is SC.WF.VQ_Decode(ich,"Dat"(it)). When the wideband data is scaled photo current or optical intensity, the VQ decoding would not be used. The payoff is in quantization error management and bit volume for the quadrature sampled data that is commonly generated for linear FM data.
Governing quality metrics	
See Also	LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) PD.HiIndx.Array(j), MD.Sensor.lidarMode SC.WF.TxProfile.Dat(il,j)

20.6 LD.* Lidar Data

LD.Point.*

LD.Point.* parameters describe lidar wide band point data representation

[253] Refers to either a receved of transmit wave form that has been VQ encoded.

20.6.1 LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) [L2]

Brief Definition

LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) gives the intensity value for points in a channel specific data block. This is principally for use at enterprise level-2 where points are dominantly channel unique with a single intensity per channel-point.

indices

The index, j, has a default range value of 1:1 (or 0:0 as implementers prefer) for the case of PD.IntTileMap.Intensity.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.Intensity.*TileNum*.Flg >0, i.e. when repetition factors are used, there are are two values in the range of j, e.g, 1:2. This index supports repetition factor use to reduce data volume.

The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the receive intensity data array in the data tag. The characters in *ChNum* take on the BCS translation of the integer values of ich, which ranges from PD.LoIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m), that is, over the channels of the current data set.

TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile.

The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

Units/representation

Dimensionless, Unsigned Integer variable length given by PD.Wordsize.Intnsy

Treatment/usage

Enterprise level-1 lidar data is time domain and never contains lidar points. At level-3 and above, we present point intensities in a data vector with an element for each channel for each point. It is the unique role of enterprise level-2 data to be able to expose and save intermediate data in the processing thread between the time domain data and the realm of points with multichannel intensities. Thus, this enterprise level-2 CMMD defines LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) as a tag for a block of point intensity data unique to the channel identified in the value of the substring, *ChNum*. Within such a block there is only one intensity per point. Effectively, the points at this stage are presented with intensities as if each channel were a separate lidar.

For linear mode and pulse compression lidar, the lidar measures the intensity as a function of fast time from which the processing can derive the intensity for a ray at a particular range. For Geiger mode (natively a non-linear sensor response), intensity for a point return, which is proportional to the reflectivity and reduced by any propagation loss is not measured by the lidar itself on a pulse basis. Thus, LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) could only be populated for Geiger mode data sets in the case of reporting synthetic lidar points. We do not choose this option for the following reason: There is a family of methods for estimating point intensity from Geiger mode data based on counting returns in voxels, but the coding options in this CMMD may not well cover the specific prescription to be reported as lidar data at the point level in a given data set generated by a processor with a synthetic point generating model. Such derived intensities are not direct physical measurements and may not only have a strong dependence on algorithm parameter values, but may or may not correspond well to scene reflectivities. To handle them appropriately, this CMMD supports reporting such intensity estimates in a user defined field [UD.PtDat.P*n*.*TileNum*(ipt)] where they will not be mistaken for lidar measurements.

Even if radiance at the receive aperture were calibratable, the conversion to intensity at the scattering point in the scene or reflectivity is impractical to determine in an absolute sense. Thus, the intensity associated with a lidar point is provided as a relative quantity. There are basically two options that will determine the range of product intensity values:

- The processor chooses a word size (from the options for PD.Wordsize.Intnsy) that is likely to be large enough, applies available deterministic corrections {SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k), SC.WF.TxProfile.Peak(il), SC.Laser.Beam(il,i,j)} to the selected peak values off the received waveform and reports them with no "normalization" (call this the natural normalization), or
- when data volume is a pressing issue, do option 1 (perhaps even using floating point calculations in a working intensity variable), search for a swath maximum and swath minimum and linearly transform the relatively corrected data to the entire linear interval of a value of PD.Wordsize.Intnsy chosen to match the range of data values. This could include a step where we take the log to achieve a desirable range of data values to be output. {Note: the log is safe because there will never be a zero intensity point.}

The enterprise level-2 processor will make a choice from the above options. Either path will generate non-integer intermediate values that will require

some form of nearest integer "interpolation'. It really does not matter which one was made. That is a quality and data set size issue. There is no exploitation impact of one over the other, although we must note the nonlinearity introduced if the log step is employed by populating ES.Point.IntensyFlg appropriately.

It is recommended that LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) never be interpreted as the *reflectivity* of scatterers at a lidar point. A future parameter might be added to this CMMD that can support conveying estimates of path attenuation that can provide the basis for attaching reflectivities to elements of (possibly synthetic) point clouds at various enterprise levels. Until the data dictionary is so extended, any reflectivities to be attached to points should be done so via a user defined point parameter.

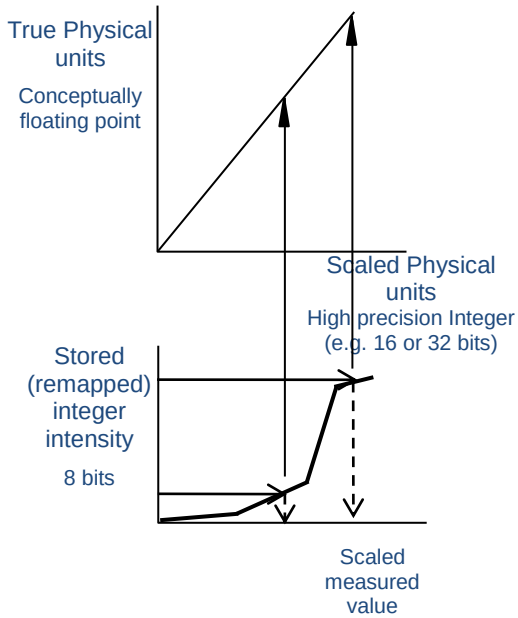
Intensities will always be real positive numbers. Since we do not expect absolute calibration to be practical for several technical generations, there is no need to provide either data as floating point numbers or floating point conversion factors to physical units.[254]

However, in the case of multichannel lidar, the ratio of channel intensities is a source of information about the scene. While absolute single channel radiometric calibration is impractical, the relative calibration of the channels is readily supported if there are truthed reflectors in the scene. The relative channel calibration is facilitated if the floating point data item, ES.Truth.IntnsyRef(ipt,ich) is populated at enterprise level-1 and used at the truth points to determine the channel calibration ratios, ES.Truth.CalChRatios(ich)[255], which can then be populated for L-2 and above. The use of the ES.Truth.CalChRatios(ich) requires that the point cloud intensity response be linear over the full dynamic range before application. Thus, it is not compatible with data stored in LD.Point.Intensity.*ChNum*.*TileNum*(ipt, j) without inverting any logarithmic or piecewise intensity remapping as described below. The appropriate L-2 wordsizing for the intensity data can support optimizing the channel to channel ratio estimation for L-3 multichannel points.

[254] If a radiometrically calibrated lidar emerges, it is a simple matter to add ES.Point.IntnsyRemapFL(j,k) as a floating conversion to physical units analog to ES.Point.IntnsyRemap(j,k).

[255] The channel calibration ratio information can also be populated a'priori from other information paths, such as laboratory measurement based calibration tables.

Depending on the dynamic range of the original determined intensities, reduction to 8 bit integer intensity could require a remapping function (ES.Point.IntsnyRemap(j,k), if populated) to be applied to this data item to accomodate available dynamic range. The diagram below indicates how measurements proportional to the true absolute intensities are reconstructed from data compressed by a piecewise linear remapping function.



See ES.Point.IntsnyRemap(j,k) for more on the piecewise linear remap function.
See ES.Point.IntensyFlg to determine whether the stored intensity values are derived from measurement linearly proportional to the incident photon flux or to its logarithm.

Governing quality metrics: Correct values populated

See Also: ES.Point.IntnsyRemap(j,k) PD.IntTileMap.Intensity.*TileNum*.Flg PD.Wordsize.Intnsy UD.PtDat.P*n*.*TileNum*(ipt) PF.SynMethod.Intensity ES.Point.IntensyFlg

20.6.2 LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) [L2]

Brief Definition: LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) supplies data that can be linearly transformed to earth frame 3-D Cartesian coordinate points in the lidar point cloud within a channel specific tile.

indices: The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the receive intensity data array in

the data tag. The characters in *ChNum* take on the BCS translation of the integer values of ich, which ranges from PD.LoIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m), that is, over the channels of the current data set.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile.

ipt is the point "row" of the data table
The index, “ipt”, is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile).

The index, j, has a default range of 1,2,3 corresponding to 3 coordinates {X,Y,Z or the code in ES.Swath.SensorBasedCoordFlg(j) if it is populated} for the case of PD.IntTileMap.Pos.*TileNum*.Flg set to zero or not populated. When PD.IntTileMap.Pos.*TileNum*.Flg =1, then j=1,2,3,4. This index supports repetition factor use to reduce data volume. The value in LD.Point.Pos.*TileNum*(ipt,1) is the repeat factor for the *tuple* LD.Point.Pos.*TileNum*(ipt,2), LD.Point.Pos.*TileNum*(ipt,3), and LD.Point.Pos.*TileNum*(ipt,4), which carry the coordinate values for a point. Note that the repetition factor support is for the tuple only and not the coordinates (e.g, X,Y or Z) separately.

Units/representation: Dimensionless, variable Integer,[256] byte length and sign type given by PD.Wordsize.Pos

Treatment/usage: LD.Point.Pos.*ChNum*.*TileNum*(ipt, j) is especially provided for enterprise level-2 data where points for channels maintain their distinct directly measured quality so that there is only one scalar intensity value for each 3-D channel-point. LD.Point.Pos.*TileNum*(ipt,j) is provided for L-3 and higher data set levels to give the point coordinate positions that have been synthesized to each have a vector of channel intensities. These separate coordinate lists are necessary so that we can keep point related data aligned with the dummy ipt index in the two cases that 1) point tiles are channel specific and when 2) points have multichannel intensities.

[256] For HDF5 transport, the DAL can provide the integer wordsize without a separate metadata item for the number of bits/bytes (because this is specifically supported by the HDF5 software). But the general case would need a metadata item that gives the wordsize if it is variable.

LD.Point.Pos.*ChNum*.*TileNum*(ipt,j) provides the principal core values that determine the relative positions of points and the shape of a point cloud.
To permit reducing the word size, the actual values of coordinates in meters (and angle units when appropriate) are obtained via a linear transformation that multiplies these dimensionless "step" numbers by a conversion factor of meters per step [ES.Scale.X_Fac, ES.Scale.Y_Fac ES.Scale.Z_Fac] and adds an offset [ES.Offset.X_Offset(j_tile), ES.Offset.Y_Offset(j_tile), ES.Offset.ZX_Offset(j_tile)].
The resultant coordinate values in meters are with respect to an earth or sensor frame per ES.Swath.SensorBasedCoordFlg(j) whose alignment and position are given in ES.Origin.* and ES.Align.*.

Governing quality metrics

The uncertainties in coordinate values are determined via a tool that performs a formal error propagation based on support data in the lidar data set. The characteristic values of the output of such a tool for the given data set are provided in MD.Qual.GeoLocUnc*. The quality metric measures the compatibility of the provided coordinate values with the error propagation process. This is a distortion figure of merit, as the linear transformation values drive absolute position fidelity. Distortion=RMS shear factor, RMS scale gradient, …

See Also

ES.Scale.X_Fac ES.Offset.X_Offset(j_tile), ES.Origin.X_ECEF(j_tile) ES.Origin.X_MGRS(j_tile) ES.Align.X_ECEF(j_tile) PD.IntTileMap.Pos.*TileNum*.Flg PD.Wordsize.Pos PD.TileMap.N_rows(j_tile) ES.Swath.SensorBasedCoordFlg(j)

20.6.3 LD.SAL.DataGrid.*ChNum*.*TileNum*(i,j,k) [L2]

Brief Definition

LD.SAL.DataGrid.*ChNum*.*TileNum*(i,j,k) is a data grid for complex pixel values due to synthetic aperture processing of lidar data.

indices

Index, i, ranges from PD.SAL.LoIndx(1) to PD.SAL.HiIndx(1), that is from the low value of the range-like direction sample number to the low value of the range-like direction sample number. If PD.SAL.LoIndx(1)=1, PD.SAL.HiIndx(1) is the number of range-like samples in the output grid block of tile *TileNum* and channel *ChNum.*

Index, j, ranges from PD.SAL.LoIndx(2) to PD.SAL.HiIndx(2), that is from the low value of the cross range-like direction sample number to the low value of the cross range-like direction sample number. If PD.SAL.LoIndx(2)=1, PD.SAL.HiIndx(2) is the number of cross range-like samples in the output grid block of tile *TileNum* and channel *ChNum.*

Index, k, =1,2 for the I and Q parts of the complex pixel amplitude.

	The evaluated character substring, *ChNum*, acts as an effective index by marking channel specific instantiations of the receive intensity data array in the data tag. The characters in *ChNum* take on the BCS translation of the integer values of ich, which ranges from PD.LoIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m) to PD.HiIndx.Channlzn.ChRg(PD.HiIndx.Channlzn.Rg_m), that is, over the channels of the current data set. *TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile.
Units/representation	Dimensionless Relative magnitude, Float
Treatment/usage	*Stub, Details pending, Need to support tree and table data in 9.7*
Governing quality metrics	
See Also	PD.SAL.HiIndx(1)

LD.WFClp.*

LD.WFClp.* parameters describe lidar wide band waveform data representation

20.6.4 LD.WFClp.RxDatVec.*TileNum*(ipi,iray, it) [L1,2]

Brief Definition	LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) carries waveform data
indices	*TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile". Use of *TileNum* for wave form clips enterprise L-2: Waveforms in enterprise L-2 data are subsets ("clips") of the full receive window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "*N*R*nnnnnn*"[257] needs to be unique. Index, ipi[258], ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*. *Special note for waveform data resulting from channel combination*: In the case that channel combination has been used to create the wave form clips at enterprise level-2, a situation akin to synthetic point generation arises and there is not necessarily a single pulse or ray that identifies a waveform vector any longer. However, we shall identify a channel combined waveform clip with a time within the temporal integration interval and a look direction within the field of view of integration referenced to that time. The pulse and

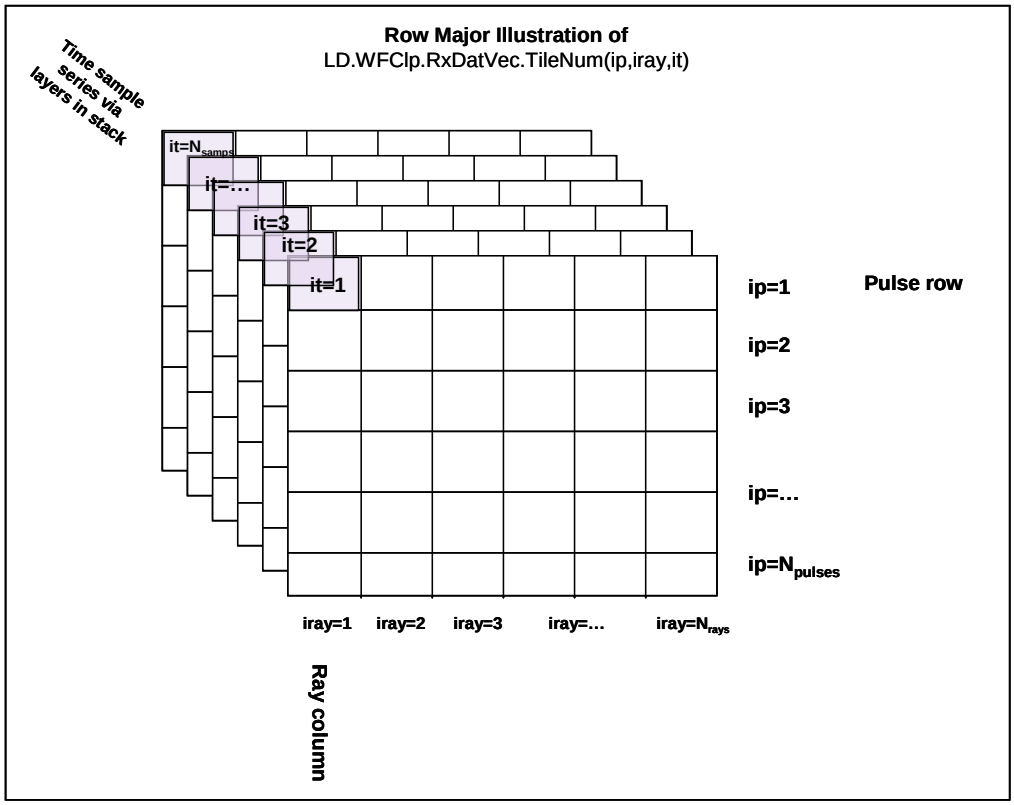
[257] Note that to be consistent with the returns coding [see for example, ES.WFClp.RtnsCode.*TileNum*(iray, ipi~~, j~~] the numerical value of *N* must be from the list, (1,2,3,4,5,6,7). However, we use N=100 to indicate a **last** return in *TileNum*, regardless of the actual return number.

[258] The index, "ipi", replaces "ip" at L-2 so that sparse pulse wave form population is possible. At L-1, every pulse wave form (RxDatVec) must be in the file so an ip that just counts IPPs will not waste storage. But only a small subset of pulses may have waveforms at L-2, so ipi is not a consecutive counter of IPPs, only of "used IPPs

	ray that most closely correspond to the time and direction attributed to the channel combined waveform clip have index values defined on a *collected* channel basis. Those are used to identify the channel combined waveform data. Thus the translation of ipi and iray index values to times and directions is exactly the same for channel combined data as for collected channel data. Only the interpretation must be tempered by the above comments and considered in light of PF.Channlzn.Hist.WF_Dat(j). Note that the output of channel combined waveform clips is only possible at enterprise level-2. Channel combined waveform clips represent both a synthetic wave form (combining the data of multiple rays and possibly pulses within single channels) and a new channelization. To be clear, after the combination process, a channel combined wave form clip will be associated with a single pre-combined basis pulse-ray. The ray (iray) will be associated with the collection channel that most closely matches the look direction for the combined data, and analogously for the pulse (ipi). Index, iray, ranges from PD.TileMap.RayRange(j_tile, 1) to PD.TileMap.RayRange(j_tile, 2), that is from the first ray in the tile to the last ray in the tile. The definitions of iray, if populated, will either be present in SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) or SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). Enterprise level-1 will tend to use the global iray definitions. Enterprise level-2 data sets will vary depending on the level-2 generating application and the determination of whether there are few enough waveforms to warrant using a small, tile oriented, waveform list approach to minimize file size. We do not use a channel index here because, as mentioned below, there is a separate channel number item for each waveform clip. See above discussion for the case of channel combined wveform clips. Index, “it”, varies from 1 to PD.TileMap.MaxNsamps(j_tile), that is, from 1 to the maximum waveform length in the tile. Each value of “it” corresponds to a separate sample of the waveform in fast time.
Units/representation	Dimensionless, variable Unsigned Integer provided by PD.Wordsize.RxDatVec
Treatment/usage	LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it) carries the fast time received data vectors, commonly called “waveforms” within the lidar community, as a data cube. There is one such cube per tile and as many tiles as desired to include all of the data in a data set[259] associated with a single collection operation. In a row major implementation, the fast time vectors themselves are the vertical

[259] At enterprise L-2, this may be only a subset of the total collection.

stacks of data formed from a common cell taken in each layer of the cube. These stacks are arranged in an array with rows asssociated with a single pulse number (IPP within which the data was measured) and columns associated with a single ray (a ray is a choice of focal plane array and a specific element).



The data cube approach to organizing the "waveforms" permits the exact same data constructs to be applied to enterprise level-1 data, which is the raw, full receive window signal for all pulses and rays generated by the lidar sensor and to enterprise level-2 waveform data, which due to processing may have only short clips of fast time from a fraction of the rays and a fraction of the pulses. In the case that the cube may be populated with waveform clips of varying length, the range of index "it" is driven by the longest of those clips in the tile, *TileNum*. The active number of samples for individual clips is found in PD.WFClp.Nsamps.*TileNum*(ip, iray).

Per the mode unification discussion in 10.1.1.3, the stack may be deep when linear mode waveforms are captured or it may consist of only a single clock latching time per quench in the case of Geiger mode.

	While the values in each word of the fast time stack are dimensionless, they need to be converted to engineering units via a process indicated by the value of MD.Sensor.lidarMode. Specifically, in Geiger mode the integer is to be interpreted as a count[260] of fast data capture clock cycles corresponding to the detector element (commonly a pixel on a APD array) latching pulse occurrence. To convert this to seconds, the fast time clock interval in seconds for array "ia" is found by (seconds per fast clock step)=SC.Swath.TimeStep/SC.Swath.RateFactor(ia) When linear mode is used, each sample in the vector is proportional to intensity and needs to be multiplied by a scaling factor to get to a consistent (on a pulse to pulse basis) record of the photo current to incident flux ratio [see SC.Pulse.ElmntSnstvty.*ChNum.TileNum*(ip,ie,k)] . All intensities in the lidar conceptual model are relative and thus they are naturally dimensionless. In the case that a form of pulse compression is employed, the data vectors are filled with vector quantized codes for fast time sampled sensor measurements. A decoding table will be provided and the each word of the range vector will act as a look up index into the decoding table. See SC.WF.VQ_Decode(ia,icode). The channel for which a fast time wave form data vector in LD.WFClp.RxDatVec.*TileNum*(ip,iray,it) was recorded is given in ES.WFClp.ChNum.*TileNum*(ipi,iray). The tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter ip starts over from 0 for each tile. The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases).

[260] This count is 0 for the fast data capture interval associated with the opening of the receive window as indicated by SC.Pulse.RecordStrt.*TileNum*(ip, j) and SC.Swath.RecordDelay.

	Care must be taken by the data set writing software to assure that the values of ipi and iray as reconstructed from repeat factor based arrays or as used in the full logical arrays in LD.WFClp.RxDatVec.*TileNum*(ipi,iray,it), PD.WFClp.*, and ES.WFClp.* all consistently map the information for a specific wave form clip to that wave form clip where a specific wave form clip at enterprise level-2 is uniquely identified by a return number, pulse number and ray identifier (recall there is one ray for an element number and array number combination). Using applications can then safely use tile specific pulse and ray values to match all waveform clip information.
Governing quality metrics	
See Also	MD.Sensor.lidarMode SC.Pulse.ElmntSnstvty.*ChNum*.*TileNum*(ip,ie,k) PD.WFClp.Nsamps.*TileNum*(ip, iray) PD.TileMap.MaxNsamps(j_tile), SC.WF.VQ_Decode(ia,icode) PD.Wordsize.RxDatVec PD.TileMap.PulseRange(j_tile,1) ES.WFClp.ChNum.*TileNum*(ipi,iray)

20.6.5 LD.WFClp.Tstart.*TileNum*(iray/irays, ipi/j) [L1,2]

Brief Definition	LD.WFClp.Tstart.*TileNum*(iray, ipi) gives the fast clock count into the receive window for the beginning of a waveform clip
indices	*TileNum*, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as an array of values in a block of data referred to by this document as a "tile". Use of *TileNum* for wave form clips enterprise L-2: Waveforms in enterprise L-2 data are subsets ("clips") of the full receive window record. Per 10.3, these **waveform tiles** **are specific to a return number.** [The nth peak, or selected data interval, in fast time order in a waveform vector is the nth "return"] Thus, the *TileNum* character string must indicate a "return" number". So labeled, all waveform clips in a single tile are of the same return number. Since *TileNum* is alpha numeric, the *TileNum* can incorporate a structure for the character substring like "1R000123" where the number before "R" is the **R**eturn number and the digits after "R" just count tiles and only the entire "*N*R*nnnnnn*"[261] needs to be unique. Default case: {PD.IntTileMap.Tstart.*TileNum*.Flg unpopulated or explicitly

[261] Note that to be consistent with the returns coding [see for example, ES.WFClp.RtnsCode.*TileNum*(iray/irays, ipi/j)] the numerical value of *N* must be from the list, (1,2,3,4,5,6,7). However, we use N=100 to indicate a **last** return in *TileNum*, regardless of the actual return number.

	set to 0} Index, iray, ranges from PD.TileMap.RayRange(j_tile, 1) to PD.TileMap.RayRange(j_tile, 2), that is, from the first ray in the tile to the last ray in the tile. The tile association of j_tile appearing in the ray range data and sub-string *TileNum* in PD.WFClp.Tstart.*TileNum*(iray, ipi) is made via PD.TileMap.TileNum(j_tile). The definitions of iray, if populated, will either be present in SC.WF.RayDesc.ArryGlbl(iray) and SC.WF.RayDesc.ElmtGlbl(iray) or SC.WF.RayDesc.Arry.*TileNum*(iray) and SC.WF.RayDesc.Elmt.*TileNum*(iray). The *TileNum* specific ray descriptions create a unique shortened list of rays with consecutive values of "iray" for each specific tile. When these are in use, "iray" runs over only the rays that have populated waveform clips in the wideband data. Enterprise level-1 will tend to use the global iray definitions. Enterprise level-2 data sets will vary depending on the level-2 generating application and the determination of whether there are few enough waveforms to warrant using a small, tile oriented, waveform list approach to minimize data volume. Index, ipi[262], ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*. The channel assignment for this waveform start time is mapped through the pulse and ray via ES.WFClp.ChNum.*TileNum*(ipi,iray). The tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter ip starts over from 0 for each tile. The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3)

[262] The index, "ipi", replaces level-1's "ip" index symbol at L-2 so that sparse pulse waveform population is possible. At L-1, every pulse waveform (RxDatVec) must be in the file; thus an ip that just counts IPPs will not waste storage. But only a small subset of pulses may have waveforms at L-2, so ipi is not a consecutive counter of IPPs, only of "used" IPPs.

	is the actual IPP count from the first IPP of the collection data set to the first IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0. Index, j_parent, is present to support generalization of PD.TileMap.StartPulseNum.*TileNum* when used for point oriented purposes in L-4 data where more than one parent lidar data set may contribute points to the cloud. In an enterprise level-2 data set the range of j_parent is 1 (0:0 or 1:1 in most cases). <u>Repetition factor in use case</u>: {PD.IntTileMap.Tstart.*TileNum*.Flg explicitly set to value greater than 0} See <u>internal tile mapping.</u> The index, iray, is replaced by a dummy row index that ranges from 1 to PD.IntTileMap.Tstart.*TileNum*.Flg, that is over the number of listed iray entries in the repetition version of the array. We call the index, "irays", in this case instead of using the default, notation "iray". Instead of identifying the ray via the index, we will use the value of the j=1 element of each data row to indicate the ray. The pulse index, "ipi", is replaced by a new dummy index, j, =1,2,3. This index supports repetition factor use to reduce data volume. Values of PD.WFClp.Tstart.*TileNum*(irays,1) are iray numbers and PD.WFClp.Tstart.*TileNum*(irays,2) are repetition factors and PD.WFClp.Tstart.*TileNum*(irays,3) will have Tstart values.
Units/representation	Dimensionless, UInt32
Treatment/usage	The lidar receive window opens up at a specific time after the transmitted pulse for each IPP. See SC.Swath.RecordDelay. The value of LD.WFClp.Tstart.*TileNum*(iray,ipi) for a specific pulse and ray is the fast time sampling clock delay after record start [itself offset SC.Swath.RecordDelay + SC.Pulse.RecordStrt.*TileNum*(ip, j) from IPP start]. This is most easily understood as the number of fast time samples of the full waveform to be discarded in skipping to the present waveform clip. That may be converted to a time via SC.Swath.TimeStep and SC.Swath.RateFactor(ia) and that time, in turn, may be transformed into a range and ultimately a Z position [see ES.WFClp.Locs.*TileNum*(ipi,iray, j)] in scene space. For enterprise level-1 data, all of the clips present are actually full receive window intensity histories. For Enterprise level-2, the typical case is that clips are histories of temporal sub-intervals (which may be associated with extended return region range extents) of the receive window. The start times for clips are typically delayed from the start of the receive window. A good strategy to reduce data volume is to capture groups of clips that have the same fast time start index and duration. LD.WFClp.Tstart.*TileNum*(iray, ipi) can convey a diversity of clip start times, as needed to characterize collected data.

Refer to the roundtrip computation diagram.

Repetition factor use case:
PD.WFClp.Tstart.*TileNum*(irays,2) is the number of times the value in PD.WFClp.Tstart.*TileNum*(irays,3) consecutively repeats in single stretch of "ipi" values corresponding to no change in wave clip start time over a range of pulses. In other words, PD.WFClp.Tstart.*TileNum*(irays,2) is the number of consecutive pulses for a ray given by PD.WFClp.Tstart.*TileNum*(irays,1) for which the value of Tstart is given in PD.WFClp.Tstart.*TileNum*(irays,3). PD.WFClp.Tstart.*TileNum*(irays,3) is the number of fast time samples in the waveform clip skipped before the first waveform sample of the waveform clip is encountered described for ray index, PD.WFClp.Tstart.*TileNum*(irays,1), in the pulse number to be recovered by accumulating the repeat factors in PD.WFClp.Tstart.*TileNum*(irays,2) per the discussion in 10.2.1.5. Any pulse-ray value not covered in the reduced table of size PD.IntTileMap.Tstart.*TileNum*.Flg by 3 is defaulted to Tstart=0. See internal tile mapping.
For illustration, consider the table below.

irays	j=1	j=2	j=3
1	iray=1	100	Tstart=1024
2	iray=1	200	Tstart=10567
3	iray=1	100	Tstart=0
4	iray=1	1000	Tstart=20444
5	iray=2	1400	Tstart=2560
6	iray=3	100	Tstart=1024
7	iray=3	200	Tstart=6008
8	iray=3	100	Tstart=12444

This set of 24 numbers replaces a 3 by 1400 array of numbers. We have a compression factor of 525. This says that:

- PD.IntTileMap.Tstart.*TileNum*.Flg=8
- the equivalent default "uncompressed" array PD.WFClp.Tstart.*TileNum*(iray, ipi) describes 3 rays and 1400 pulses and had

PD.WFClp.Tstart.*TileNum*(1, 1 through 100) =1024
PD.WFClp.Tstart.*TileNum*(1, 101 through 300) =10567
PD.WFClp.Tstart.*TileNum*(1, 301 through 400) =0
PD.WFClp.Tstart.*TileNum*(1, 401 through 1400)=20444
PD.WFClp.Tstart.*TileNum*(2, 1 through 1400)=2560
PD.WFClp.Tstart.*TileNum*(3, 1 through 100) =1024
PD.WFClp.Tstart.*TileNum*(3, 101 through 300) =6008

	PD.WFClp.Tstart.*TileNum*(3, 301 through 400) =12444 PD.WFClp.Tstart.*TileNum*(3, 401 through 1400) =0 The first 8 statements above come from accumulating the repeat factors in PD.WFClp.Tstart.*TileNum*(irays, j). The last line follows from the default rule that items not defined in the table have value zero. Note that we had to embed Tstart=0 for PD.WFClp.Tstart.*TileNum*(1,301 through 400). This assures correct assessment of the pulse number by accumulation for pulses that follow the Tstart=0 pulse stretch. Note that in principle, a delay of 0 after the start of a receive window is physically well defined, although not of much interest. So we need to refer to PD.WFClp.Nsamps.*TileNum*(iray, ip) to be certain of whether there is an actual waveform clip starting at Tstart=0 fast time sample intervals after the first interval of the receive window.
Governing quality metrics	
See Also	PD.TileMap.TileNum(j_tile) PD.IntTileMap.Tstart.*TileNum*.Flg SC.Swath.TimeStep SC.Swath.RateFactor(ia) SC.Swath.RecordDelay round trip time computation in SC.Pulse.TxDelay.*TileNum*(ip) ES.WFClp.Locs.*TileNum*(ipi,iray, j) ES.WFClp.ChNum.*TileNum*(ipi,iray) SC.Pulse.RecordStrt.*TileNum*(ip, j)

20.7 UD.* User Defined Wide Band Data

UD.* parameters describe how user defined wideband items not explicitly defined in this CMMD may be added in compliant developmental and interim implementations. The UD.* metadata items act to enable a data set to self define data in the presence of data items yet to be added to the CMMD, but nevertheless of particular interest in a setting that uses modifications of operational software modules for creating and manipulating sensor data sets. In this way, the CMMD establishes a standardization that can enhance innovation and rapid technology insertion rather than stifling such advances in operational practice.

20.7.1 UD.PtDat.P*n.TileNum*(ipt) [L2,3,4,5]

Brief Definition	UD.PtDat.P*n.TileNum*(ipt) carries point oriented wide band user defined data.
indices	The index, "ipt", is a dummy that indicates the data row to which data in a vertical tile vector belong. Refer to Appendix 10.2.1 and, in particular, figures 10.2.1-1 and 10.2.1-3. The number of rows in each tile is provided in PD.TileMap.N_rows(j_tile). *TileNum*, though part of a tag name, acts somewhat as an index. It indicates

	which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile. The evaluated sub-string "*n*" also behaves somewhat as an index. A separate value of *n* is used for each user defined wide band point oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two wideband data items. These might be, for example, UD.PtDat.P1.*TileNum*(ipt) and UD.PtDat.P2.*TileNum*(ipt).
Units/representation	Variable, see PD.UserDefined.PtType.P*n*
Treatment/usage	UD.PtDat.P*n*.*TileNum*(ipt) is one of 3 groups of data fields for user defined data. Each value of the evaluated string, *n*, indicates a distinct user defined point oriented wide band data parameter. Each UD.PtDat.P*n*.*TileNum*(ipt) item, with evaluated *n* field, is given its actual definition in the text string which is the value of PD.UserDefined.PtDef.P*n*. The purpose of providing such user defined data fields is to provide flexibility to special user needs and to avoid the problem of minor variations in weighting, terms, filter sizes leading to a proliferation of entities/tags or compromised interoperability via near miss definition issues. An example of a common user defined varable for L-2 use is slope variance, which could have any one of many similar but different definitions. Similarly, a user might define a variable to indicate the likelihood that a point is signal or that it is noise, for example.
Governing quality metrics	
See Also	PD.UserDefined.PtType.P*n* PD.UserDefined.PtDef.P*n*

20.7.2 UD.PulseDat.P*n*.*TileNum*(ipi) [L1,2,3,4,5]

Brief Definition

UD.PulseDat.P*n*.*TileNum*(ipi) carries pulse oriented wide band user defined data.

indices

Index, ipi, ranges from PD.LoIndx.WFClpPulseNum.*TileNum* to PD.HiIndx.WFClpPulseNum.*TileNum*, that is over the entries of the list of pulses for which waveform clips appear in the return specific tile, *TileNum*.[262,259]

The tile level translation from ipi to ip is given by PD.WFClp.PulseNum.*TileNum*(ipi)=ip, where the pulse counter ip starts over from 0 for each tile. The term, "tile level", indicates that to find the actual pulse number referenced to the first pulse of the collected data from which the current data set was generated, one must use IPP number = "tile level" ip + PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) where PD.TileMap.StartPulseNum.*TileNum*(j_parent,PD.LoIndx.Tsegts.*TileNum*,3) is the actual IPP count from the first IPP of the collection data set to the first

IPP of the first intra-tile time segment of wide band data in the tile. That IPP will be locally indexed as 0.
TileNum, though part of a tag name, acts somewhat as an index. It indicates which uniquely named parameter carries the data as a vector of values in a block of data referred to by this document as a Tile.

The evaluated sub-string "*n*" also behaves somewhat as an index. A separate value of *n* is used for each user defined wide band point oriented parameter. So, in the case where a user has definitions for a first and second wide band parameter, the data set would contain two wideband data items. These might be, for example, UD.PulseDat.P1.*TileNum*(ipt) and UD.PulseDat.P2.*TileNum*(ipt).

Units/representation: Variable, see PD.UserDefined.PulseType.P*n*

Treatment/usage: UD.PulseDat.P*n*.*TileNum*(ipi) is one of 3 groups of data fields for user defined data. Each value of the evaluated string, *n*, indicates a distinct user defined pulse oriented wide band data parameter. Each UD.PulseDat.P*n*.*TileNum*(ipi) item, with evaluated *n* field, is given its actual definition in the text string which is the value of PD.UserDefined.PulseDef.P*n*. The purpose of providing such user defined data fields is to provide flexibility to special user needs and to avoid the problem of minor variations in weighting, terms, filter sizes leading to a proliferation of entities/tags or compromise interoperability via near miss definition issues.

Governing quality metrics

See Also: PD.UserDefined.PulseType.P*n* PD.UserDefined.PulseDef.P*n*

20.7.3 *Eventually* UD.WaveDat.Pn.TileNum(ip,iray) [L1,2]

Brief Definition: UD.WaveDat.P*n*.*TileNum*(ip,iray) carries wave clip oriented wide band user defined data. This is a stub for future

indices

Units/representation

Treatment/usage

Governing quality metrics

See Also

Annex A, Acronyms

APD	Avalanche Photon Detector
API	Application Program Interface
BCS	Basic Character Set
CSM	Community Sensor Model
DAL	Data Access Library
DDMS	DoD Discovery Metadata Standard
DIAL	Differential Absorption lidar
DIGEST	Digital Geographic Information Exchange Standard
DISR	DoD IT Standards Registry
DNI	Director of National Intelligence
DoD	Department of Defense (US)
DR	Discrepancy Report
ENU	East North Up
FOV	Field of View
GEOINT	Geospatial Intelligence
GM	Geiger Mode
GML	Geography Markup Language
GPAS	Geospatial Publicly Available Standards
GPS	Global Positioning System
GSIP	GEOINT Structure Implementation Profile
GWG	GEOINT Standards Working Group
HRE	High Resolution Elevation Data
IC	Intelligence Community (US)
IERS	International Earth Rotation and Reference Systems Service
IPP	InterPulse Period
ISO TC	International Organisation for Standardisation Technical Committee
IT	Information Technology
ISR	Intelligence Surveillance Reconnaissance
Kmz	Keyhole markup zipped
LIDAR	Light Detection and Ranging
LM	Linear Mode
LOS	Line of Sight
MCP	Micro Channel Plate
MFG	Metadata Focus Group
MGRS	Military Grid Reference System
NED	North East Down
NGA	National Geospatial-Intelligence Agency
NMF	National System for Geospatial Intelligence Metadata Foundation
NMIS	NSG Metadata Implementation Specification

NSG	National System for Geospatial Intelligence (US)
O&M	Operations and Maintenance
OGC	Open Geospatial Consortium
PED	Processing Exploitation Dissemination (architecture)
PRF	Pulse Repetition Frequency
RSS	Root Sum (of) Squares
SGC	Sensor Geopositioning Center
SI	International System of Units (Systeme Internationale)
TJD	Truncated Julian Day
US	United States
VRP	Volume Return Product
WTM	World Aeronautical Chart (**W**AC) **T**arget **M**osaic

Annex B, Sample Profile

The following spreadsheet lists the names for data items in an example CMMD compliant data set.

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
PD List				
PD	LoIndx	AdjacentP(j)		
PD	HiIndx	AdjacentP(j)		
PD	LoIndx	AdjacentW(j)		
PD	HiIndx	AdjacentW(j)		
PD	LoIndx	Array(j)		
PD	HiIndx	Array(j)		
PD	LoIndx	Array.Elem(ia)		
PD	HiIndx	Array.Elem(ia)		
PD	LoIndx	Artifacts		
PD	HiIndx	Artifacts		
PD	LoIndx	ChannIzn.ChRg(m)		
PD	HiIndx	ChannIzn.ChRg(m)		
PD	LoIndx	ChannIzn.Rg_m		
PD	HiIndx	ChannIzn.Rg_m		
PD	LoIndx	DAL		
PD	HiIndx	DAL		
PD	LoIndx	Fellows		
PD	HiIndx	Fellows		
PD	LoIndx	IPP(j)		
PD	HiIndx	IPP(j)		
PD	LoIndx	Laser(j)		
PD	HiIndx	Laser(j)		
PD	LoIndx	MD_Strings(j)		
PD	HiIndx	MD_Strings(j)		
PD	LoIndx	MD_Tree(j)		
PD	HiIndx	MD_Tree(j)		
PD	LoIndx	Overlap(j)		
PD	HiIndx	Overlap(j)		
PD	LoIndx	Parents		
PD	HiIndx	Parents		
PD	LoIndx	Point.RangeListC.01		
PD	HiIndx	Point.RangeListC.01		
PD	LoIndx	Point.RangeListX.01		
PD	HiIndx	Point.RangeListX.01		
PD	LoIndx	Point.RangeListC.02		
PD	HiIndx	Point.RangeListC.02		
PD	LoIndx	Point.RangeListX.02		
PD	HiIndx	Point.RangeListX.02		
PD	LoIndx	Point.SigmaDat		
PD	HiIndx	Point.SigmaDat		
PD	LoIndx	Point.Tile		
PD	HiIndx	Point.Tile		
PD	LoIndx	PrevStages(k)		
PD	HiIndx	PrevStages(k)		
PD	LoIndx	Pulse		
PD	HiIndx	Pulse		
PD	LoIndx	Quench		
PD	HiIndx	Quench		
PD	LoIndx	Ray.*01*		
PD	HiIndx	Ray.*01*		
PD	LoIndx	Ray.*02*		
PD	HiIndx	Ray.*02*		
PD	LoIndx	Ray.*03*		
PD	HiIndx	Ray.*03*		
PD	LoIndx	Recon(j)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
PD	HiIndx	Recon(j)		
PD	LoIndx	Remap		
PD	HiIndx	Remap		
PD	LoIndx	RgAmb(j)		
PD	HiIndx	RgAmb(j)		
PD	LoIndx	Scan(j)		
PD	HiIndx	Scan(j)		
PD	LoIndx	Swath(j)		
PD	HiIndx	Swath(j)		
PD	LoIndx	TileMap.DedictdTile		
PD	HiIndx	TileMap.DedictdTile		
PD	LoIndx	Tsegts.01		
PD	HiIndx	Tsegts.01		
PD	LoIndx	Tsegts.02		
PD	HiIndx	Tsegts.02		
PD	LoIndx	Tsegts.03		
PD	HiIndx	Tsegts.03		
PD	LoIndx	Tsegts.04		
PD	HiIndx	Tsegts.04		
PD	LoIndx	Tsegts.05		
PD	HiIndx	Tsegts.05		
PD	LoIndx	Truth(j)		
PD	HiIndx	Truth(j)		
PD	LoIndx	Voids(j)		
PD	HiIndx	Voids(j)		
PD	LoIndx	Zones		
PD	HiIndx	Zones		
PD	IntTileMap	PulseNum.01.Flg		
PD	IntTileMap	ArrayNum.01.Flg		
PD	IntTileMap	PulseNum.02.Flg		
PD	IntTileMap	ArrayNum.02.Flg		
PD	IntTileMap	PulseNum.03.Flg		
PD	IntTileMap	ArrayNum.03.Flg		
PD	IntTileMap	PulseNum.04.Flg		
PD	IntTileMap	ArrayNum.04.Flg		
PD	IntTileMap	PulseNum.5.Flg		
PD	IntTileMap	ArrayNum.5.Flg		
PD	IntTileMap	ChNum.01 .Flg		
PD	IntTileMap	ChNum.02.Flg		
PD	IntTileMap	ChNum.03.Flg		
PD	IntTileMap	ChNum.04.Flg		
PD	IntTileMap	ChNum.05.Flg		
PD	IntTileMap	DirCase.01.Flg		
PD	IntTileMap	DirCase.02.Flg		
PD	IntTileMap	DirCase.03.Flg		
PD	IntTileMap	DirCase.04.Flg		
PD	IntTileMap	DirCase.05.Flg		
PD	IntTileMap	FilterCode.01.Flg		
PD	IntTileMap	FilterCode.02.Flg		
PD	IntTileMap	FilterCode.03.Flg		
PD	IntTileMap	FilterCode.04.Flg		
PD	IntTileMap	FilterCode.05.Flg		
PD	IntTileMap	Scan.Num.06.Flg		
PD	IntTileMap	Scan.Num.07.Flg		
PD	IntTileMap	Scan.Num.08.Flg		
PD	IntTileMap	Scan.Num.09.Flg		
PD	IntTileMap	Scan.Num.10.Flg		
PD	IntTileMap	Scan.Phase.06.Flg		
PD	IntTileMap	Scan.Phase.07.Flg		
PD	IntTileMap	Scan.Phase.08.Flg		
PD	IntTileMap	Scan.Phase.09.Flg		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
PD	IntTileMap	Scan.Phase.10.Flg		
PD	IntTileMap	NadirClass.01.Flg		
PD	IntTileMap	NadirClass.02.Flg		
PD	IntTileMap	NadirClass.01.Reps(j)		
PD	IntTileMap	NadirClass.02.Reps(j)		
PD	IntTileMap	TruthTags.01.Flg		
PD	IntTileMap	TruthTags.02.Flg		
PD	IntTileMap	TruthTags.03.Flg		
PD	IntTileMap	TruthTags.04.Flg		
PD	IntTileMap	TruthTags.05.Flg		
PD	IntTileMap	TxCount.06.Flg		
PD	IntTileMap	TxCount.07.Flg		
PD	IntTileMap	TxCount.08.Flg		
PD	IntTileMap	TxCount.09.Flg		
PD	IntTileMap	TxCount.10.Flg		
PD	Point	RangeListC.1(j,k)		
PD	Point	RangeListC.2(j,k)		
PD	Point	RangeListC.3(j,k)		
PD	Point	RangeListX.1(j,k)		
PD	Point	RangeListX.2(j,k)		
PD	Point	RangeListX.3(j,k)		
PD	Sort	NativePointOrder		
PD	Sort	NativePointPri(j)		
PD	Sort	OrderFlag(j_tile)		
PD	Sort	OrderVec.01(j)		
PD	Sort	OrderVec.02(j)		
PD	Sort	OrderVec.03(j)		
PD	Sort	OrderVec.04(j)		
PD	Sort	OrderVec.05(j)		
PD	TileMap	AdjacencyPts(n-tile)		
PD	TileMap	ChanList(j_tile,k)		
PD	TileMap	DedictdTile.Cnstrmt1.Name		
PD	TileMap	DedictdTile.Cnstrmt1.Val(j)		
PD	TileMap	DedictdTile.TileNumValHi(j)		
PD	TileMap	DedictdTile.TileNumValLow(j)		
PD	TileMap	GlblPtTyping		
PD	TileMap	Lat(j_tile,k)		
PD	TileMap	Lon (j_tile,k)		
PD	TileMap	N_Rows(j_tile)		
PD	TileMap	N_Tiles		
PD	TileMap	PtType.01		
PD	TileMap	PtType.02		
PD	TileMap	PtType.03		
PD	TileMap	PtType.04		
PD	TileMap	PtType.05		
PD	TileMap	PulseRange(j_tile,k)		
PD	TileMap	RayRange(j_tile,k)		
PD	TileMap	StartPulseNum.01(j_parent, j,k)		
PD	TileMap	StartPulseNum.02(j_parent, j,k)		
PD	TileMap	StartPulseNum.03(j_parent, j,k)		
PD	TileMap	StartPulseNum.04(j_parent, j,k)		
PD	TileMap	StartPulseNum.05(j_parent, j,k)		
PD	TileMap	TileNum(j_tile)		
PD	TileMap	MGRS_Zone(j_tile)		
PD	TileMap	X_Range(j_tile,k)		
PD	TileMap	Y_Range(j_tile,k)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
PD	TileMap	Z_Range(j_tile,k)		
PD	TileMap	T_Range(j_tile,k)		
PD	UserDefined	PtDef.P1		
PD	UserDefined	PtDef.P2		
PD	UserDefined	PtType.P1		
PD	UserDefined	PtType.P2		
PD	Wordsize	Iray		
PD	Wordsize	Meta(j)		
PD	Wordsize	Pos		
PD	Wordsize	Pulse		
PD	Wordsize	PtDef.P1		
PD	Wordsize	PtDef.P2		
PD	Wordsize	Recon		
PD	Wordsize	RxDatVec		
PD	Wordsize	SensorPos		
PD	Wordsize	TileNum		
PD	Wordsize	Truth(j)		
PD	Wordsize	TxDat		
PD	Wordsize	Zones		
MD List				
MD	Aggregate	Fellow.CollctnDate(j)		
MD	Aggregate	Fellow.CollctnOpID(j)		
MD	Aggregate	Fellow.DS_Name(j)		
MD	Aggregate	Fellow.PlatformID(j)		
MD	Aggregate	Fellow.ProdType(j)		
MD	Aggregate	Fellow.SensorID(j)		
MD	Aggregate	Fellow.SensorType(j)		
MD	Aggregate	Fellow.StartTime(j)		
MD	Aggregate	Parent.CollctnDate(j)		
MD	Aggregate	Parent.CollctnOpID(j)		
MD	Aggregate	Parent.DS_Name(j)		
MD	Aggregate	Parent.Duration(j)		
MD	Aggregate	Parent.StartTime(j)		
MD	Collect	Duration		
MD	Coverage	.BE_NumList(j)		
MD	Coverage	Bounds.LatNorth		
MD	Coverage	Bounds.LatSouth		
MD	Coverage	Bounds.LonEast		
MD	Coverage	Bounds.LonWest		
MD	Coverage	Bounds.OuterZoneList(j)		
MD	Coverage	Bounds.ZoneRanges(j		
MD	Coverage	Bounds.Vertices(iv,j)		
MD	Coverage	Bounds.Z(j)		
MD	Coverage	Bounds.ZoneList(j)		
MD	Coverage	Centroid		
MD	Coverage	Cntry.Cd2(j)		
MD	Coverage	Cntry.Cd3(j)		
MD	Coverage	Temporal.Duration		
MD	Coverage	Temporal.StartTime		
MD	Coverage	Temporal.StartDate		
MD	Coverage	Voids.UnColl(j,k)		
MD	Coverage	Voids.Incompl(j,k)		
MD	Coverage	Voids.Masked(j,k)		
MD	Coverage	Voids.Atmsphrc(j,k)		
MD	File	Class		
MD	Meta	Class		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
MD	Meta	DAL_VersionName(j)		
MD	Meta	DR_ContactAddr		
MD	Meta	DS_Name		
MD	Meta	ID		
MD	Meta	Language		
MD	Meta	MaintOrg		
MD	NomGeo	Pt.Az.Ctr		
MD	NomGeo	Pt.Az.Spread		
MD	NomGeo	Pt.Nadir.Ctr		
MD	NomGeo	Pt.Nadir.Spread		
MD	NomGeo	Swath.Az.Ctr		
MD	NomGeo	Swath.Az.Spread		
MD	NomGeo	Swath.Nadir.Ctr		
MD	NomGeo	Swath.Nadir.Spread		
MD	Objective	Abstract		
MD	Objective	BE_NumList(j)		
MD	Objective	Conops(j)		
MD	Objective	DateAcceptable		
MD	Objective	DateDue		
MD	Objective	Desc		
MD	Objective	EastBound		
MD	Objective	ID		
MD	Objective	NorthBound		
MD	Objective	SouthBound		
MD	Objective	Tasker		
MD	Objective	Vertices(iv,j)		
MD	Objective	WAC_List(j)		
MD	Objective	WBStyle		
MD	Objective	WestBound		
MD	Platform	Desc		
MD	Platform	ID		
MD	ProcHist	CodeList(j)		
MD	ProcHist	L1Writer.Name		
MD	ProcHist	L1Writer.ProcDate		
MD	ProcHist	L1Writer.ProcSite		
MD	ProcHist	L1Writer.RelDate		
MD	ProcHist	L1Writer.Vendor		
MD	ProcHist	L1Writer.Versn		
MD	ProcHist	L2App.Name(j)		
MD	ProcHist	L2App.ProcDate(j)		
MD	ProcHist	L2App.ProcSite(j)		
MD	ProcHist	L2App.RelDate(j)		
MD	ProcHist	L2App.Vendor(j)		
MD	ProcHist	L2App.Versn(j)		
MD	ProcHist	L2NumStages		
MD	ProcHist	ProdType(j)		
MD	Qual	GeoLocUncH		
MD	Qual	GeoLocUncV		
MD	Qual	GSD		
MD	Qual	Impairments(j)		
MD	Qual	RetPtDensity		
MD	Range	Max		
MD	Range	Min		
MD	Range	NomRgToScene		
MD	Sensor	ID		
MD	Sensor	Lidar Mode		
MD	Sensor	LidarType		
MD	Truth	Dat(j)		
MD	Truth	Descr		
ES List				
ES	Align	LocG_Vert(it,j)		
ES	Align	X_ECEF(j_tile,j)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
ES	Align	XYZ_Flag		
ES	Align	Y_ECEF(j_tile,j)		
ES	Align	Z_ECEF(j_tile,j)		
ES	CollCondn	Obscurant		
ES	CollCondn	PercentAreaAffected		
ES	Offset	X(j_tile)		
ES	Offset	Y(j_tile)		
ES	Offset	Z(j_tile)		
ES.	Origin	GeodtcEpoch		
ES	Origin	MGRS_Flag		
ES	Origin	X_ECEF(j_tile)		
ES	Origin	X_MGRS(j_tile)		
ES	Origin	Y_ECEF(j_tile)		
ES	Origin	Y_MGRS(j_tile)		
ES	Origin	Z_ECEF(j_tile)		
ES	Origin	Z_MGRS(j_tile)		
ES	Point	FilterScore.01(ipt)		
ES	Point	FilterScore.02(ipt)		
ES	Point	FilterScore.03(ipt)		
ES	Point	FilterScore.04(ipt)		
ES	Point	FilterScore.05(ipt)		
ES	Point	PulseNum.01(ipt)		
ES	Point	PulseNum.02(ipt)		
ES	Point	PulseNum.03(ipt)		
ES	Point	PulseNum.04(ipt)		
ES	Point	PulseNum.05(ipt)		
ES	Point	SigmaXYZ(j,k)		
ES	Point	TruthTags.*TileNum* (ipt,j)		
ES	Scale	X_Fac		
ES	Scale	Y_Fac		
ES	Scale	Z_Fac		
ES	Scan	DeΠ_Cov		
ES	Scan	DecorrTimeAlpha(is)		
ES	Scan	DecorrTimeBeta(is)		
ES	Scan	DecorrTimeGamma(is)		
ES	Scan	DecorrTimeX(is)		
ES	Scan	DecorrTimeY(is)		
ES	Scan	DecorrTimeZ(is)		
ES	Scan	GM_SigmaR(j)		
ES	Scan	RhoAlphaGamma(is)		
ES	Scan	Rho XBeta(is)		
ES	Scan	RhoAlphaBeta(is)		
ES	Scan	RhoBetaGamma(is)		
ES	Scan	RhoXAlpha(is)		
ES	Scan	RhoXGamma(is)		
ES	Scan	RhoXY(is)		
ES	Scan	RhoXZ(is)		
ES	Scan	RhoYAlpha(is)		
ES	Scan	RhoYBeta(is)		
ES	Scan	RhoYGamma(is)		
ES	Scan	RhoYZ(is)		
ES	Scan	RhoZAlpha(is)		
ES	Scan	RhoZBeta(is)		
ES	Scan	RhoZGamma(is)		
ES	Scan	SigmaAlpha(is)		
ES	Scan	SigmaBeta(is)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
ES	Scan	SigmaGamma(is)		
ES	Scan	SigmaX(is)		
ES	Scan	SigmaY(is)		
ES	Scan	SigmaZ(is)		
ES	Swath	DeIT_Cov		
ES	Swath	DecorrTimeAlpha		
ES	Swath	DecorrTimeBeta		
ES	Swath	DecorrTimeGamma		
ES	Swath	DecorrTimeX		
ES	Swath	DecorrTimeY		
ES	Swath	DecorrTimeZ		
ES	Swath	GM_SigmaT		
ES	Swath	RhoAlphaGamma		
ES	Swath	Rho XBeta		
ES	Swath	RhoAlphaBeta		
ES	Swath	RhoBetaGamma		
ES	Swath	RhoXAlpha		
ES	Swath	RhoXGamma		
ES	Swath	RhoXY		
ES	Swath	RhoXZ		
ES	Swath	RhoYAlpha		
ES	Swath	RhoYBeta		
ES	Swath	RhoYGamma		
ES	Swath	RhoYZ		
ES	Swath	RhoZAlpha		
ES	Swath	RhoZBeta		
ES	Swath	RhoZGamma		
ES	Swath	SigmaAlpha		
ES	Swath	SigmaBeta		
ES	Swath	SigmaGamma		
ES	Swath	SigmaX		
ES	Swath	SigmaY		
ES	Swath	SigmaZ		
ES	Swath	SensorBasedCoordFlg(j)		
ES	Swath	PSF(j)		
ES	Truth	CalChRatios(ich)		
ES	Truth	IntnsyRef(itp,ich,j)		
ES	Truth	IllusKMZ.*n*(nchar)		
ES	Truth	CovCorrs(itp, j)		
ES	Truth	CovSigs(itp, j)		
ES	Truth	Pos(itp,j)		
PF List				
PF	Adjustmts	Artifacts(j)		
PF	Adjustmts	Scan(is,j)		
PF	Adjustmts	Subswath(iswl,isw,,j)		
PF	Channlzn	Descr(j)		
PF	Channlzn	Disp(j)		
PF	Channlzn	Hist.ChTrace(j,k)		
PF	Channlzn	Hist.Comb.Code(j)		
PF	Channlzn	Hist.Comb.Func(k)		
PF	CommonSigma	AggLevel		
PF	Point	IrayNum.01(ipt)		
PF	Point	IrayNum.02(ipt)		
PF	Point	IrayNum.03(ipt)		
PF	Point	IrayNum.04(ipt)		
PF	Point	IrayNum.05(ipt)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
PF	Point	SynTasymtry.01(ipt)		
PF	Point	SynTasymtry.02(ipt)		
PF	Point	SynTasymtry.03(ipt)		
PF	Point	SynTasymtry.04(ipt)		
PF	Point	SynTasymtry.05(ipt)		
PF	Point	SynTspread.01(ipt)		
PF	Point	SynTspread.02(ipt)		
PF	Point	SynTspread.03(ipt)		
PF	Point	SynTspread.04(ipt)		
PF	Point	SynTspread.05(ipt)		
PF	Point	SynXspread.01(ipt)		
PF	Point	SynXspread.02(ipt)		
PF	Point	SynXspread.03(ipt)		
PF	Point	SynXspread.04(ipt)		
PF	Point	SynXspread.05(ipt)		
PF	Point	SynYspread.01(ipt)		
PF	Point	SynYspread.02(ipt)		
PF	Point	SynYspread.03(ipt)		
PF	Point	SynYspread.04(ipt)		
PF	Point	SynYspread.05(ipt)		
PF	Point	SynZspread.01(ipt)		
PF	Point	SynZspread.02(ipt)		
PF	Point	SynZspread.03(ipt)		
PF	Point	SynZspread.04(ipt)		
PF	Point	SynZspread.05(ipt)		
PF	Recon	APP(j)		
PF	Recon	APP_Descr(j)		
PF	Recon	CommandFiles(j)		
PF	Recon	PrevProdNames(mm,k)		
PF	Recon	RootDS(j)		
PF	Recon	Site(j)		
PF	Registn	PulseDat(j),		
PF	Registn	ScanDat(j),		
PF	Registn	SubswathDat(j),		
PF	Sort	PointOrder.01		
PF	Sort	PointOrder.02		
PF	Sort	PointOrder.03		
PF	Sort	PointOrder.04		
PF	Sort	PointOrder.05		
PF	Sort	PointPriority.01(j)		
PF	Sort	PointPriority.02(j)		
PF	Sort	PointPriority.03(j)		
PF	Sort	PointPriority.04(j)		
PF	Sort	PointPriority.05(j)		
PF	Swath	OffsetStatus		
PF	Swath	SynBspread		
PF	Swath	SynPspread		
PF	Swath	SynTasymtry		
PF	Swath	SynTspread		
PF	Swath	SynXspread		
PF	Swath	SynYspread		
PF	Swath	SynZspread		
PF	SynMethod	Intensity		
PF	SynMethod	Position		
PF	SynMethod	Time		
PF	TimeToRange	BendEffect		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
PF	TimeToRange	Model		
PF	TimeToRange	Parms.Ht_Surface		
PF	TimeToRange	Parms.Hu_Surface		
PF	TimeToRange	Parms.Lat		
PF	TimeToRange	Parms.Nsurface		
PF	TimeToRange	Parms.P_Surface		
PF	TimeToRange	Parms.ScaleHeight(j)		
PF	TimeToRange	Parms.T_Surface		
SC List				
SC	Array	BadElmnts.*ia*(j,k)		
SC	Array	FilterDat(ia,j)		
SC	Array	OptclCent(j)		
SC	Array	PolDat(ifc,j)		
SC	Array	RayDirs.*Ia*(icase,ie,,j)		
SC	Array	RayDirsSigmas(ia,icase,,j)		
SC	Array	CurrentResp.Dat(ia,j)		
SC	Array	CurrentResp.Flg(ia)		
SC	Array	LatticeDesc(ia,j)		
SC	Artifacts	List(j)		
SC	Artifacts	Descr(j)		
SC	Channlzn	Descr(j)		
SC	Channlzn	Disp(j)		
SC	Laser	Beam(i,j)		
SC	Laser	Dat(il,j)		
SC	Laser	DelCosAlpha		
SC	Laser	DelCosBeta		
SC	Laser	IDat(il,k)		
SC	MultiDwell	Flg		
SC	MultiDwell	Dat(j)		
SC	Overlap	ScanPrcnt		
SC	Overlap	Scan.ID1(j)		
SC	Overlap	Scan.ID2(j)		
SC	Overlap	Scan.X_Hi(j)		
SC	Overlap	Scan.X_Lo(j)		
SC	Overlap	Scan.Y_Hi(j)		
SC	Overlap	Scan.Y_Lo(j)		
SC	Overlap	SubswathPrcnt		
SC	Pulse	ArrayMajorAxis.06(j, ip)		
SC	Pulse	ArrayMajorAxis.07(j, ip)		
SC	Pulse	ArrayMajorAxis.08(j, ip)		
SC	Pulse	ArrayMajorAxis.09(j, ip)		
SC	Pulse	ArrayMajorAxis.10(j, ip)		
SC	Pulse	ArrayMinorAxis.06(j, ip)		
SC	Pulse	ArrayMinorAxis.07(j, ip)		
SC	Pulse	ArrayMinorAxis.08(j, ip)		
SC	Pulse	ArrayMinorAxis.09(j, ip)		
SC	Pulse	ArrayMinorAxis.10(j, ip)		
SC	Pulse	BmMajorAxis.06(j, ip)		
SC	Pulse	BmMajorAxis.07(j, ip)		
SC	Pulse	BmMajorAxis.08(j, ip)		
SC	Pulse	BmMajorAxis.09(j, ip)		
SC	Pulse	BmMajorAxis.10(j, ip)		
SC	Pulse	BmMinorAxis.06(j, ip)		
SC	Pulse	BmMinorAxis.07(j, ip)		
SC	Pulse	BmMinorAxis.08(j, ip)		
SC	Pulse	BmMinorAxis.09(j, ip)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
SC	Pulse	BmMinorAxis.10(j, ip)		
SC	Pulse	FilterCode.06(ip)		
SC	Pulse	FilterCode.07(ip)		
SC	Pulse	FilterCode.08(ip)		
SC	Pulse	FilterCode.09(ip)		
SC	Pulse	FilterCode.10(ip)		
SC	Pulse	IPP_DurSeq.01(ipn)		
SC	Pulse	Pos.X_Offset(j_tile)		
SC	Pulse	Pos.Y_Offset(j_tile)		
SC	Pulse	Pos.Z_Offset(j_tile)		
SC	Pulse	Scan.Num.06(ip)		
SC	Pulse	Scan.Num.07(ip)		
SC	Pulse	Scan.Num.08(ip)		
SC	Pulse	Scan.Num.09(ip)		
SC	Pulse	Scan.Num.10(ip)		
SC	Pulse	Scan.Phase.06(ip.j)		
SC	Pulse	Scan.Phase.07(ip.j)		
SC	Pulse	Scan.Phase08(ip.j)		
SC	Pulse	Scan.Phase.09(ip.j)		
SC	Pulse	Scan.Phase.10(ip.j)		
SC	Pulse	SensorAxisX.06(ip, j)		
SC	Pulse	SensorAxisX.07(ip, j)		
SC	Pulse	SensorAxisX.08(ip, j)		
SC	Pulse	SensorAxisX.09(ip, j)		
SC	Pulse	SensorAxisX.10(ip, j)		
SC	Pulse	SensorAxisY.06(ip, j)		
SC	Pulse	SensorAxisY.07(ip, j)		
SC	Pulse	SensorAxisY.08(ip, j)		
SC	Pulse	SensorAxisY.09(ip, j)		
SC	Pulse	SensorAxisY.10(ip, j)		
SC	Pulse	SensorPos.01.01(j,ip)		
SC	Pulse	SensorPos.01.02(j,ip)		
SC	Pulse	SensorPos.01.03(j,ip)		
SC	Pulse	SensorPos.01.04(j,ip)		
SC	Pulse	SensorPos.01.05(j,ip)		
SC	Pulse	SensorPos.02.01(j,ip)		
SC	Pulse	SensorPos.02.02(j,ip)		
SC	Pulse	SensorPos.02.03(j,ip)		
SC	Pulse	SensorPos.02.04(j,ip)		
SC	Pulse	SensorPos.02.05(j,ip)		
SC	Pulse	TxCount.01.01(ip)		
SC	Pulse	TxCount.01.02(ip)		
SC	Pulse	TxCount.01.03(ip)		
SC	Pulse	TxCount.01.04(ip)		
SC	Pulse	TxCount.01.05(ip)		
SC	Pulse	TxCount.02.01(ip)		
SC	Pulse	TxCount.02.02(ip)		
SC	Pulse	TxCount.02.03(ip)		
SC	Pulse	TxCount.02.04(ip)		
SC	Pulse	TxCount.02.05(ip)		
SC	Pulse	TxDelay.*06*(ip)		
SC	Pulse	TxDelay.*07*(ip)		
SC	Pulse	TxDelay.*08*(ip)		
SC	Pulse	TxDelay.*09*(ip)		
SC	Pulse	TxDelay.*10*(ip)		
SC	RgAmb	StratFlg(j)		

Sample Profile---a posssible population list for L2 data and metadata				inherited data
Class	**Property**	**trailing name detail**		**augmented**
SC	RgAmb	Num.01(iscanNadirClass,iphase).		
SC	RgAmb	Num.02(iscanNadirClass,iphase).		
SC	Scan	Axes.ThetaMaj		
SC	Scan	Axes.ThetaMin		
SC	Scan	ConePulseTable(is)		
SC	Scan	N_Major(is,j)		
SC	Scan	NomGroundLen		
SC	Scan	NadirClass.01.(is)		
SC	Scan	NadirClass.02.(is)		
SC	Scan	NomNadir.Mean(isnc)		
SC	Scan	NomNadir.Min(isnc)		
SC	Scan	NomNadir.Max(isnc)		
SC	Scan	PhaseBins(isnc,j)		
SC	Scan	Type		
SC	Scan	U_Cone(is,j)		
SC	Swath	Pos.X_Fac		
SC	Swath	Pos.Y_Fac		
SC	Swath	Pos.Z_Fac		
SC	Swath	RateFactor(ia)		
SC	Swath	RecordDelay		
SC	Swath	Timestep		
SC	Swath	TimeSync		
SC	WF	RayDescArryGlbl(iray)		
SC	WF	RayDescElmtGlbl(iray)		
SC	WF	Rx.Nsamps(ia)		
LD List				
LD	Point	Pos.01.01(ipt,j)		
LD	Point	Pos.01.02(ipt,j)		
LD	Point	Pos.01.03(ipt,j)		
LD	Point	Pos.01.04(ipt,j)		
LD	Point	Pos.01.05(ipt,j)		
LD	Point	Pos.02.01(ipt,j)		
LD	Point	Pos.02.01(ipt,j)		
LD	Point	Pos.02.01(ipt,j)		
LD	Point	Pos.02.01(ipt,j)		
LD	Point	Pos.02.01(ipt,j)		
UD List				
UD	PtDat	P1.01(ipt)		
UD	PtDat	P1.02(ipt)		
UD	PtDat	P1.03(ipt)		
UD	PtDat	P1.04(ipt)		
UD	PtDat	P1.05(ipt)		
UD	PtDat	P2.01(ipt)		
UD	PtDat	P2.02(ipt)		
UD	PtDat	P2.03(ipt)		
UD	PtDat	P2.04(ipt)		
UD	PtDat	P2.05(ipt)		